

# Unix Services

## Capitolo 9

9. Unix Services	pag. 197
1. System Boot	pag. 198
1. Il sistema init systemd	pag. 198
2. Il sistema init System V	pag. 204
2. Remote Login	pag. 207
1. Secure Remote Login: SSH	pag. 207
1. Autenticazione basata su chiave	pag. 208
2. Come da usare da remoto le applicazioni X11	pag. 210
3. Port Forwarding: come creare gli Encrypted Tunnels	pag. 210
2. Come utilizzare i Remote Graphical Desktops	pag. 211
3. Gestione dei diritti [permessi]	pag. 213
4. Interfacce di amministrazione	pag. 215
1. Come amministrare attraverso un'interfaccia web: webmin	pag. 215
2. Configurazione dei pacchetti: debconf	pag. 217
5. Eventi di sistema di syslog	pag. 218
1. Principi e funzionamento	pag. 218
2. Il file di configurazione	pag. 219
1. Sintassi del Selector	pag. 219
2. Sintassi delle Actions	pag. 219
6. Linetd Super-Server	pag. 220
7. Scheduling Tasks attraverso cron e atd	pag. 221
1. Formato di un file crontab	pag. 222
2. Come utilizzare il comando at	pag. 224
8. Scheduling Asynchronous Tasks: anacron	pag. 225
9. Quotas	pag. 225
10. Backup	pag. 227
1. Backup con rsync	pag. 227
2. Ripristino delle macchine senza Backups	pag. 229
11. Hot Plugging: hotplug	pag. 230
1. Introduzione	pag. 230
2. Il problema dell'assegnazione dei nomi	pag. 230
3. Come funziona udev	pag. 230
4. Un caso concreto	pag. 232
12. Power Management: Advanced Configuration and Power Interface (ACPI)	pag. 234

<<Questo capitolo tratta una serie di servizi base, che sono diffusi su molti sistemi Unix. Tutti gli amministratori dovrebbero averne familiarità>>

## 9.1. System Boot

[Lo "scrolling" in informatica e nella produzione cinematografica e televisiva si riferisce allo scorrimento orizzontale o verticale di testo, immagini o video su uno schermo, display o monitor.]

Durante il boot del computer, sulla console si susseguono molti messaggi sotto forma di testo scorrevole [scrolling] che si riferiscono alle inizializzazioni ed alle configurazioni automatiche in corso di esecuzione. Per poter modificare, anche lievemente, l'esecuzione di questa fase, dovete conoscerla a fondo. Questo capitolo intende pertanto trattare l'argomento a questo scopo.

Inizialmente, il BIOS prende il controllo del computer, rileva i dischi, carica il Master Boot Record ed esegue il bootloader. Dopodiché subentra il bootloader che rileva, carica ed esegue il kernel sul disco. Inizializzato il kernel, incomincia il rilevamento della partizione contenente il root filesystem, così che la summenzionata partizione possa essere montata e possa essere avviato infine il primo programma: `init`. Di norma la "root partition" ed il programma `init` sono in concreto collocati su un virtual filesystem che esiste solo nella RAM (da cui deriva il nome dello schema di caricamento del temporaneo root filesystem ovvero `initramfs`, in passato denominato `initrd` da `initialization RAM disk`). Il suddetto filesystem temporaneo viene caricato nella memoria dal bootloader, spesso da un file su un disco rigido o sulla rete. Inoltre il filesystem temporaneo contiene il minimo indispensabile al kernel per caricare "il vero" root filesystem ossia: i driver modules per i dischi rigidi o per altri dispositivi senza i quali il sistema non potrebbe essere avviato; oppure, nella maggior parte dei casi, i moduli e gli scripts di inizializzazione per assemblare i RAID arrays [gli arrays molto genericamente sono delle "strutture dati" mentre il termine RAID è l'acronimo di "Redundant Array of Inexpensive Disks" o di "Redundant Array of Independent Disks"], per aprire le partizioni crittografate, per attivare i volumi LVM, ecc. ... Una volta montata la root partition, `initramfs` cede il controllo "al vero" `init` e la macchina ritorna al processo di boot standard.

### 9.1.1 Il sistema init systemd

Il "vero init" è al momento supportato da `systemd` e questo paragrafo si occupa proprio di questo `init` system.

<b>CULTURA</b> Prima dell'avvento di <code>systemd</code>	<code>systemd</code> è un "init system" relativamente recente e, sebbene fosse già disponibile con Wheezy (in parte), è solo a partire da Jessie che è stato utilizzato di default. I rilasci antecedenti di Debian usavano per impostazione predefinita il "System V init" (incluso nel pacchetto <code>sysv-rc</code> ), un sistema [di boot] molto più tradizionale. Il System V verrà trattato più avanti.
<b>ALTERNATIVA</b> Altri boot systems	<p>Questo libro descrive sia il boot system utilizzato di default da Debian Buster (implementato dal pacchetto <code>systemd</code>), sia il vecchio sistema (a suo tempo di default) <code>sysvinit</code>, che è a sua volta deriva ed è erede dei sistemi System V Unix; in ogni caso esistono altri sistemi.</p> <p><code>file-rc</code> è un boot system con un processo semplificato. Il <code>file-rc</code> mette in atto il criterio dei <code>runlevels</code>, ma attraverso un singolo file di configurazione che specifica ad <code>init</code> i processi che devono essere avviati e l'ordine con cui devono essere lanciati, sostituisce le directories ed i collegamenti simbolici. [Molto genericamente il <code>runlevel</code> indica lo stato di attività di una macchina (e dei suoi processi in esecuzione) a cui viene associato, attraverso un indice numerico che va da 0 a 6, un "livello".]</p> <p>Lo <code>upstart</code> system, al momento della stesura di questo libro non è ancora stato del tutto testato su Debian. Si tratta di un event based; [un evento è un'azione o una circostanza riconosciuta e gestita dal software, spesso di provenienza esterna, che può essere stata generata o attivata dal sistema, dall'utente, ecc.] gli <code>init</code> scripts non vengono più eseguiti in sequenza ma in risposta ad eventi come ad esempio il completamento di altri scripts dai quali dipendono. Questo sistema, inizialmente disponibile solo su Ubuntu, è stato poi incluso anche in Debian Jessie, ma non come sistema predefinito: ha semplicemente sostituito <code>sysvinit</code> e difatti una delle tasks lanciate da <code>upstart</code> si occupa dell'avvio degli scripts scritti per i sistemi tradizionali, in particolare quelli del pacchetto <code>sysv-rc</code>.</p> <p>Esistono molti altri sistemi ed altre operating modes, come <code>runit</code> o <code>minit</code>, ma sono abbastanza circoscritti e non molto diffusi.</p>

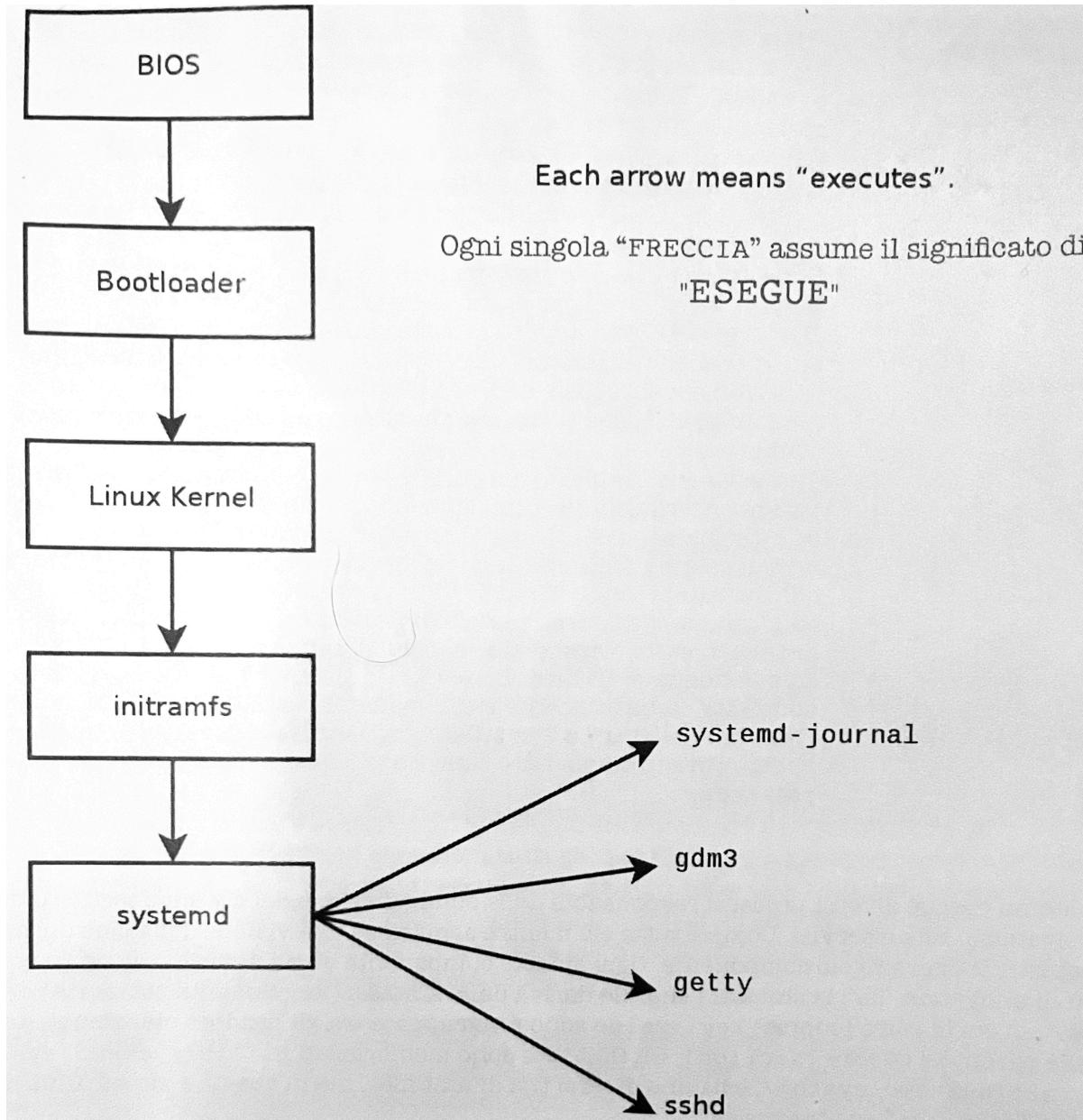


Figura 9.1 Boot sequence di un computer che sta eseguendo Linux con systemd.

#### SPECIFIC CASE Booting dalla rete

In alcune configurazioni, il BIOS potrebbe non essere stato configurato per eseguire l'MBR, bensì per recuperarne l'equivalente dalla rete, in modo da poter "mettere insieme" un computer anche se privi di un disco rigido oppure una macchina in grado di reinstallarsi completamente ad ogni avvio. Questa opportunità non è compatibile con tutti i tipi di hardware e necessita generalmente di una configurazione adeguata del BIOS e della scheda di rete. L'avvio dalla rete può essere utilizzato per avviare debian-installer o FAI (andate a leggere il paragrafo 4.1, "Metodi di Installazione" a pag. 52).

#### BASILARE Il processo, una program instance

Un processo è la rappresentazione in memoria di un programma in esecuzione. Riunisce tutte le informazioni necessarie per il corretto funzionamento di un software (il suo stesso codice, ma anche i dati che ha in memoria, l'elenco dei files che ha aperto, le connessioni di rete che ha stabilito, ecc.). Un singolo programma potrebbe essere "istanziato" in numerosi processi non necessariamente in esecuzione sotto differenti user IDs.

[Molto genericamente in computer science l'espressione "istanziare" viene utilizzata ogni volta che un "context" (un insieme di dati che se salvato consente l'interruzione ed il riavvio di una "task") viene creato in base a dei modelli "riferiti" (chiamati in causa indirettamente attraverso un valore) ovvero si dice che "i modelli sono stati istanziati". L'espressione istanziare viene utilizzata anche per la programmazione orientata agli oggetti. Per una comprensione più approfondita e, magari più corretta, fare riferimento alla manualistica].

## SICUREZZA

Come ottenere attraverso una shell aperta con un'opzione init i diritti di root

Il primo processo avviato per convenzione è il programma `init` (che per impostazione predefinita è un collegamento simbolico a `/lib/systemd/systemd`). Tuttavia è possibile trasmettere un'opzione `init` al kernel che indichi un altro programma.

Ovvero chiunque abbia accesso fisico al computer potrà, premendo il tasto Reset, riavviarlo e tramite il prompt del bootloader, trasmettere l'opzione `init=/bin/sh` al kernel per ottenere l'accesso root pur non conoscendo la password dell'amministratore.

Onde evitare ciò, dovrete impostare una password per il bootloader. Ricordatevi di proteggere anche con una password l'accesso al BIOS (a patto che il vostro BIOS supporti un meccanismo di protezione con una password), altrimenti un malintenzionato potrà sempre avviare la macchina da un removable media [usb, dvd, ecc.] contenente un sistema Linux personalizzato, che utilizzerà per accedere ai dati sui dischi rigidi del computer.

Si precisa inoltre che la maggior parte dei BIOSes è accessibile attraverso una password generica. Inizialmente questo sistema era stato designato come soluzione per coloro i quali avevano dimenticato la loro password, ma ormai queste passwords sono pubbliche e distribuite su Internet (potrete verificarlo cercando "generic BIOS passwords" con un motore di ricerca). In conclusione tutti questi metodi di protezione rallenteranno i tentativi di accesso non autorizzati alla macchina, senza essere in grado di prevenirli completamente. Ovvero non esiste un sistema in grado di proteggere un computer da un attacco fisico: il malintenzionato può difatti smontare i dischi rigidi per collegarli ad un altro computer di cui ne ha il controllo, o persino rubare l'intero computer o cancellare la memoria del BIOS per resettare la password ...

Systemd esegue diversi processi responsabili della configurazione del sistema: tastiera, drivers, filesystems, rete e servizi. Compie tutto ciò mentre acquisisce una visione d'insieme del sistema e dei requisiti di ogni singolo componente. Ogni singolo componente viene descritto da un (o qualche volta più di uno) "unit file"; la sintassi generale deriva dalla sintassi (ampiamente utilizzata) dei "\*.ini files", in cui le pairs (coppie) `key = value` sono raggruppate tra gli headers menzionati attraverso delle parentesi quadre `[section]`. Gli Unit files sono memorizzati in `/lib/systemd/system/` ed in `/etc/systemd/system/`; esistono diversi tipi di unit files, ma in questa sede saranno analizzati solo i "services" ed i "targets".

Un systemd "service file" descrive un processo gestito da systemd. Contiene all'incirca le stesse informazioni dei vecchi init-scripts, ma espresse in modo dichiarativo (e molto più conciso).

Systemd si occupa della maggior parte delle repetitive tasks (ovvero dell'avvio e dell'arresto del processo, della verifica dello status, del logging [la registrazione dei logs], del dropping privileges [trad. lett. "decadenza dei privilegi"], ecc.). Ed il "service file" si limita solo ad elencare le specifiche del processo. Segue un esempio di "service file" per SSH:

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmenFile=-/etc/default/ssh
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
```

```
[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Come potrete notare non c'è molto codice nel file utilizzato come esempio, solo dichiarazioni. Systemd si occupa della visualizzazione dei reports sullo stato di avanzamento, tiene traccia dei processi e li riavvia se necessario.

Un systemd "target file" descrive uno stato del sistema, in cui una serie di servizi è risaputo sono attivi. Il "target file" può essere immaginato come un succedaneo del vecchio "runlevel". Uno dei suddetti "targets" è local-fs.target; quando questo "stato del sistema" viene raggiunto, il resto del sistema può presumere che tutti i filesystems locali sono stati montati e sono accessibili. Esistono altri targets tra cui network-online.target e sound.target. Le dipendenze di un target file possono essere elencate nello stesso target file (attraverso una riga con Requires=) oppure utilizzando un collegamento simbolico al service file desiderato nella directory del target desiderato /lib/systemd/system/targetname.target.wants/. Ad esempio, /etc/systemd/system/printer.target.wants/ contiene un collegamento a /lib/systemd/system/cups.service; systemd pertanto si assicurerà che CUPS venga eseguito per poter raggiungere lo stato di sistema delineato da printer.target.

Dato che gli unit files sono per lo più dichiarativi anziché essere scripts o programmi, non possono essere eseguiti direttamente, bensì devono essere interpretati da systemd. Per tale ragione diverse utilities consentono all'amministratore di interagire con systemd e di controllare lo stato del sistema e di ogni singolo componente.

La prima utility che occorre citare è systemctl. Se eseguirete systemctl privo di qualsiasi "arguments" (trad. lett. "argomento"), vi elencherà tutti gli unit files noti a systemd (eccetto gli unit files che sono stati disabilitati) ed il loro stato. systemctl status offre una migliore presa-visione dei servizi e dei processi correlati. Se fornirete a systemctl il nome di un servizio (ad esempio systemctl status ntp.service), vi restituirà ancora più dettagli, nonché le ultime (poche) log lines relative al servizio (tratteremo meglio questo tema più avanti).

Potrete avviare manualmente un servizio attraverso systemctl semplicemente con systemctl start servicename.service. Diversamente potrete arrestare un servizio con systemctl stop servicename.service; esistono altri sottocomandi, come reload (trad. lett. "carica nuovamente") e restart (trad. lett. "riavvia").

Per attivare un servizio (in modo che si avvii automaticamente all'avvio del computer) utilizzate systemctl enable servicename.service, mentre per disabilitarlo systemctl disable servicename.service. Per verificare lo stato del servizio utilizzate systemctl is-enable servicename.service.

Un'altra caratteristica interessante di systemd è che include una logging component denominata journald. journald può essere utilizzato in aggiunta ai tradizionali sistemi di logging come syslogd: aggiunge funzionalità interessanti come l'associazione di ogni singolo messaggio con il servizio che lo ha generato e la possibilità di acquisirne i messaggi di errore generati dall'initialization sequence. I messaggi possono essere visualizzati in seguito attraverso il comando journalctl. Il comando journalctl se utilizzato senza argomenti mostra semplicemente tutti i log messages che sono stati registrati dal system boot; e raramente viene impiegato in questo modo. Il più delle volte, il suddetto comando viene utilizzato con un service identifier [del servizio di cui si vogliono vedere i log messages]:

```
# journalctl -u ssh.service
-- Logs begin at Tue 2015-03-31 10:08:49 CEST, end at TUE 2015-03-31 17:06:02
CEST.
-> --
Mar 31 10:08:55 mirtuel sshd[430]: Server listening on 0.0.0.0 port 22.
Mar 31 10:08:55 mirtuel sshd[430]: Server listening on :: port 22.
Mar 31 10:09:00 mirtuel sshd[430]: Received SIGHUP; restarting
```

```

Mar 31 10:09:00 mirtuel sshd[430]: Server listening on 0.0.0.0 port 22.
Mar 31 10:09:00 mirtuel sshd[430]: Server listening on :: port 22.
Mar 31 10:09:32 mirtuel sshd[1151]: Accepted password for roland from 192.168.1.129
    -> port 53394 ssh2
Mar 31 10:09:32 mirtuel sshd[1151]: pam_unix(sshd:session): session opened for user
    -> roland by (uid=0)

```

Un'altra utile "command-line flag" è `-f`, che indica a `journalctl` di visualizzare i nuovi messaggi mentre vengono emessi (allo stesso modo di come fa `tail -f file`).

Se vi sembra che un servizio non funzioni correttamente, il primo passo per risolvere il problema è verificarne la corrente esecuzione con `systemctl status`; qualora i messaggi visualizzati da questo primo comando non fossero sufficienti per identificare il problema, è consigliabile consultare i messaggi raccolti da `journald`. Ad esempio, presumiamo che il server SSH non funzioni:

```

# systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
  Active: failed (Result: start-limit) since Tue 2015-03-31 17:30:36 CEST; 1s ago Process:
1023 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS) Process: 1188
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited, status=255)
  Main PID: 1188 (code=exited, status=255)

Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
    -> status=255/n/a
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service start request repeated too quickly,
    -> refusing to start.
Mar 31 17:30:36 mirtuel systemd[1]: Failed to start OpenBSD Secure Shell server.
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered failed state.

# journalctl -u ssh.service
-- Logs begin at Tue 2015-03-31 17:29:27 CEST, end at Tue 2015-03-31 17:30:36 CEST.
-- 
Mar 31 17:29:27 mirtuel sshd[424]: Server listening on 0.0.0.0 port 22.
Mar 31 17:29:27 mirtuel sshd[424]: Server listening on :: port 22.
Mar 31 17:29:29 mirtuel sshd[424]: Received SIGHUP; restarting.
Mar 31 17:29:29 mirtuel sshd[424]: Server listening on 0.0.0.0 port 22.
Mar 31 17:29:29 mirtuel sshd[424]: Server listening on :: port 22.
Mar 31 17:30:10 mirtuel sshd[1147]: Accepted password for roland from 192.168.1.129
    -> port 38742 ssh2
Mar 31 17:30:10 mirtuel sshd[1147]: pam_unix(sshd:session): session opened for user
    -> roland by (uid=0)
Mar 31 17:30:35 mirtuel sshd[1180]: /etc/ssh/sshd_config line 28: unsupported option
    -> "yess".
Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
    -> status=255/n/a
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:35 mirtuel sshd[1182]: /etc/ssh/sshd_config line 28: unsupported option
    -> "yess".

```

```

Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
--> status=255/n/a
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:35 mirtuel sshd[1184]: /etc/ssh/sshd_config line 28: unsupported option
--> "yess".
Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
--> status=255/n/a
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:36 mirtuel sshd[1186]: /etc/ssh/sshd_config line 28: unsupported option
--> "yess".
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
--> status=255/n/a
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:36 mirtuel sshd[1188]: /etc/ssh/sshd_config line 28: unsupported option
--> "yess".
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exited, code=exited,
--> status=255/n/a
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered failed state.
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service start request repeated too quickly,
--> refusing to start.
Mar 31 17:30:36 mirtuel systemd[1]: Failed to start OpenBSD Secure Shell server.
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered failed state.
# vi /etc/ssh/sshd_config
# systemctl start ssh.service
# systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
  Active: active (running) since Tue 2015-03-31 17:31:09 CEST; 2s ago
    Process: 1023 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
   Main PID: 1222 (sshd)
     CGroup: /system.slice/ssh.service
             └─1222 /usr/sbin/sshd -D
#

```

Nell'esempio soprastante dopo la verifica dello stato del servizio ("failed"), sono stati esaminati i logs, che hanno indicato un errore nel file di configurazione. Sempre nel suddetto esempio: è stato modificato il file di configurazione per correggere l'errore; è stato riavviato il servizio e verificato nuovamente che funzionasse (con esito positivo).

<b>ANDANDO OLTRE</b> Altri tipi di unit files	<p>In questo paragrafo sono stati descritti gli aspetti più basilari di <code>systemd</code>. In realtà possiede molte altre funzionalità; qui ne verranno elencate solo alcune:</p> <ul style="list-style-type: none"> <li>• socket activation: un “socket” unit file descrive un network socket o un Unix socket gestito da <code>systemd</code>; concretamente il socket verrà creato da <code>systemd</code> ed il servizio subordinato potrà essere avviato su richiesta in conseguenza ad un tentativo di connessione. Approssimativamente replica la feature set [trad. lett. “l’insieme di funzionalità”] di <code>inetd</code>. Andate a leggere <code>systemd.socket(5)</code>.</li> <li>• timers: un “timer” unit file descrive degli eventi che si verificano ad intervalli regolari o in un determinato momento; quando un servizio è collegato ad un timer, la task corrispondente verrà eseguita ogni qual volta scatta il timer. In parte replica le funzionalità supportate da <code>cron</code>. Andate a leggere <code>systemd.timer(5)</code></li> <li>• network: un “network” unit file definisce un’interfaccia di rete, consentendo di configurarla e confermando che il servizio dipende dal fatto che una specifica interfaccia sia attiva.</li> </ul>
--------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 9.1.2. Il sistema init System V

Il sistema init System V (ivi definito meramente `init`) esegue un intero set di processi in base alle istruzioni contenute nel file `/etc/inittab`. Il primo programma che esegue (corrispondente alla fase `sysinit`) è `/etc/init.d/rcS`, uno script che a sua volta esegue tutti i programmi contenuti dalla directory `/etc/rcS.d/`.

Tra questi programmi ci sono quelli designati:

- alla configurazione della tastiera della console;
- al caricamento dei drivers: la maggior parte dei moduli del kernel sono caricati dallo stesso kernel in base all’hardware rilevato; i drivers aggiuntivi saranno quindi caricati automaticamente, se i corrispondenti moduli saranno elencati in `/etc/modules`;
- alla verifica dell’integrità dei filesystems;
- al montaggio di partizioni locali;
- alla configurazione della rete;
- al montaggio di network filesystem (NFS).

<b>BASILARE</b> Moduli Kernel e opzioni	<p>I moduli del kernel hanno anche delle opzioni che potrete configurare inserendo i files in <code>/etc/modprobe.d/</code>. Le suddette opzioni sono definite attraverso delle direttive come ad esempio: <code>options module-name option-name=option-value</code>. Potrete specificare diverse opzioni con un’unica direttiva, se necessario. I summenzionati files di configurazione sono stati concegnati per <code>modprobe</code> – il programma vi consentirà di caricare un modulo del kernel con le sue dipendenze (i moduli possono infatti chiamare altri moduli). Questo programma viene distribuito attraverso <code>kmod</code>.</p>
--------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dopo questa fase, `init` prende il controllo ed avvia i programmi associati al runlevel predefinito (solitamente il runlevel 2). Esegue `/etc/init.d/rc 2`, script che avvia tutti i servizi elencati in `/`

`etc/rc2.d/` che iniziano con la lettera "S". Storicamente i due numeri susseguiti il carattere alfabetico indicavano la posizione nell'ordine d'avvio dei servizi, ma al giorno d'oggi il boot system predefinito utilizza `insserv`, che pianifica l'esecuzione dei servizi in base alle loro dipendenze. Di conseguenza ciascun boot script dichiara le condizioni in base alle quali occorre avviare o arrestare un servizio (ad esempio se un servizio deve essere avviato prima o dopo un altro servizio); `init` pertanto li avvia in un ordine che soddisfi le suddette condizioni. Inoltre la numerazione statica degli scripts non è più presa in considerazione (anche se ancora oggi la loro nomenclatura impone che debbano iniziare con la lettera "s" seguita da due numeri susseguiti e dall'effettivo nome dello script utilizzato per le dipendenze). Generalmente dapprima vengono avviati i servizi fondamentali (come ad esempio il logging attraverso `rsyslog` o il port assignment attraverso `portmap`) e successivamente i servizi standard e l'interfaccia grafica (`gdm3`).

Il suddetto boot system basato sulle dipendenze automatizza il re-numbering [la re-assegnazione di numeri nominali ai servizi per determinarne l'ordine d'avvio], che potrebbe risultare noioso se lo si dovesse eseguire manualmente, limitando di fatto l'errore umano, in quanto la pianificazione viene eseguita in base a dei parametri dichiarati. Inoltre consente l'avvio parallelo di più servizi, se sono indipendenti l'uno dall'altro, potenzialmente accelerando il processo di boot.

`init` scompartisce i diversi runlevels [in base a dei criteri], in modo che possa eseguirne lo switch l'uno dall'altro attraverso il comando `telinit new-level`. Immediatamente, dopo l'immissione di questo comando, `init` esegue di nuovo `/etc/init.d/rc` con il nuovo runlevel desiderato. Questo script a sua volta avvia i "missing service" [trad. lett. "servizi mancanti"] ed arresta quelli che non sono più desiderati. Per fare ciò, fa riferimento al contenuto della directory `/etc/rcx.d` (dove "x" rappresenta il nuovo livello di esecuzione). Gli scripts il cui nome inizia con la lettera "S" (che sta per "Start") sono i servizi che devono essere avviati, quelli che iniziano con "K" (che sta per "Kill") sono i servizi che devono essere arrestati. Questo tipo di script evita che vengano riavviati i servizi attivi negli antecedenti runlevels.

In Debian, il sistema init System V utilizza per impostazione predefinita solo quattro diversi runlevels:

- Level 0 viene utilizzato provvisoriamente, durante la fase di arresto del computer (powering down). Di conseguenza contiene molti scripts "K".
- Level 1, noto anche come `single-user-mode`, corrisponde al sistema in `degraded mode` [modalità che si attiva a seguito di un malfunzionamento di un RAID]; include solo i servizi-base ed è stato designato per operazioni di manutenzione in cui le interazioni con gli utenti di basso profilo non sono consentite.
- Level 2 è il normale livello operativo, che include servizi di rete, interfaccia grafica, user logins, ecc.
- Level 6 è simile al livello 0, viene utilizzato durante la fase di shutdown (spegnimento) che precede un reboot (riavvio).

Esistono altri livelli, nello specifico dal 3° [Level 3] al 5° [Level 5]. Per impostazione predefinita, sono configurati per funzionare allo stesso modo del Level 2, ma l'amministratore può modificarli (aggiungendo o rimuovendo scripts nelle corrispondenti directory `/etc/rcx.d/`) per personalizzarli in base ad esigenze particolari.

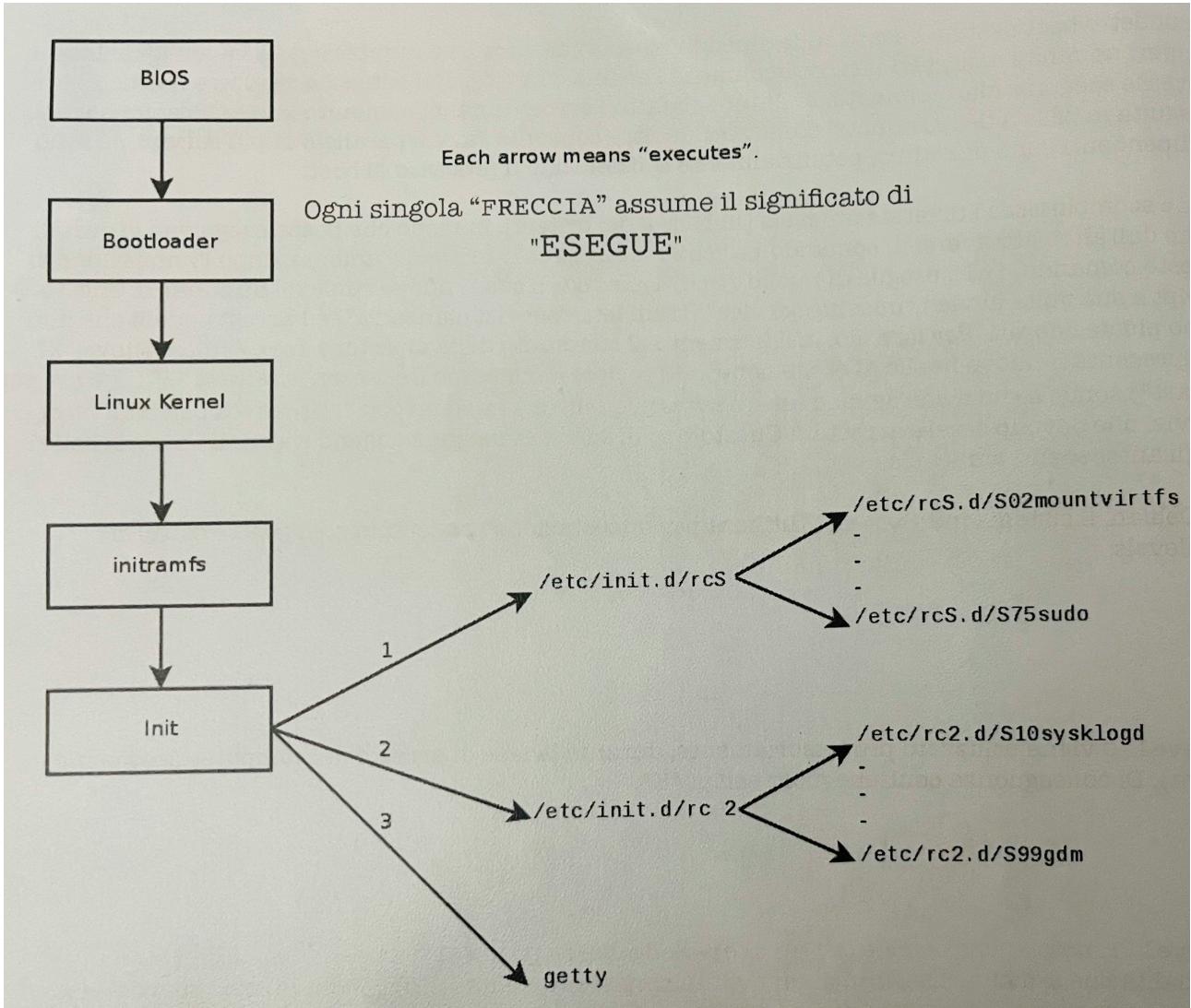


Figura 9.2 Boot sequence di un computer che sta eseguendo Linux con un sistema init Sistem V

Tutti gli scripts inclusi nelle diverse directories `/etc/rcX.d` sono di fatto solo collegamenti simbolici, creati durante l'installazione del pacchetto interessato attraverso il programma `update-rc.d` e che puntano agli scripts effettivi memorizzati in `/etc/init.d/`. L'amministratore potrà personalizzare i servizi di ciascun runlevel eseguendo nuovamente `update-rc.d`, con i parametri appropriati. La man page `update-rc.d(1)` ne descrive dettagliatamente la sintassi. Occorre precisare che rimuovere tutti i collegamenti simbolici (con il parametro `remove`) non è il metodo giusto per disattivare un servizio. Dovrete solo configurarlo per non farlo avviare durante l'esecuzione dei runlevel desiderati (preservando le chiamate correlate al suo arresto qualora il servizio venga eseguito nell'antecedente runlevel). Dato che `update-rc.d` possiede un'interfaccia piuttosto complessa, potrete utilizzare in sua alternativa `rcconf` (dal pacchetto `rcconf`) in quanto offre un'interfaccia più user-friendly.

#### DEBIAN POLICY

##### Riavvio dei servizi

I maintainer scripts [gli scripts di configurazione] dei pacchetti Debian a volte riavviano determinati servizi per garantire la disponibilità o per includere alcune opzioni. Il comando per gestire un servizio - `service` `operation` - non prende in considerazione il runlevel, presumendo (erroneamente) che il servizio sia correttamente in uso e che pertanto possa avviare operazioni inadeguate (come l'avvio di un servizio che è stato deliberatamente arrestato, oppure l'arresto di un servizio che è già stato arrestato, ecc.). Debian ha quindi introdotto il programma `invoke-rc.d`: dovrà essere usato attraverso i maintainer scripts per eseguire gli scripts di inizializzazione dei servizi (`invoke-rc.d` si limiterà ad eseguire solo i comandi necessari). Occorre evidenziare che, differentemente dalla prassi, qui il suffisso `.d` viene utilizzato nel nome di un programma e non nel nome di una directory.

Infine, init avvia i control programs per le varie virtual console (getty). Viene visualizzato un prompt, che per avviare una sessione-utente rimane in attesa di un username per poi eseguire login user.

[Genericamente un control program è un programma che svolge: operazioni di input/output; caricamento di altri programmi; rilevamento di errori; comunicazione con l'operatore e così via. Persino un sistema operativo a grandi linee è un esempio di control program].

#### DIZIONARIO

Console e terminale

I primi computers erano generalmente separati in diverse parti abbastanza ingombranti: lo storage enclosure [il box o in taluni casi l'armadio esterno per gli hard disks] e la central processing unit [cpu] erano separati dalle periferiche utilizzate dagli operatori per controllarli. Queste costituivano un accessorio di complemento a sua volta separato, la "console". Questo termine è rimasto ancora oggi in uso, ma il suo significato è cambiato. È diventato più o meno sinonimo di "terminale": essendo di fatto un set composto da una tastiera ed uno schermo. Con l'evolversi dell'informatica, i sistemi operativi hanno iniziato a supportare più console virtuali contemporaneamente per garantire più sessioni indipendenti, nonostante la presenza di una sola tastiera fisica e di un solo schermo. La maggior parte dei sistemi GNU/Linux supporta di conseguenza sei console virtuali (in text mode), accessibili grazie alle combinazioni di tasti: Control + Alt + F1 per la prima virtual console e così via sino a Control + Alt + F6 per la sesta virtual console. I termini "console" e "terminale" possono anche, in generale, riferirsi ad un emulatore di terminale in una sessione grafica X11 (ad esempio xterm, gnome-terminal o konsole).

## 9.2. Remote Login

È essenziale che un amministratore sia in grado di connettersi ad un computer da remoto. I servers, confinati in una stanza apposita, raramente sono equipaggiati di una tastiera e di uno schermo – ma sono collegati alla rete.

#### BASILARE

Client, server

Un sistema in cui diversi processi comunicano tra loro viene spesso definito utilizzando la metafora "client/server" [se tradurrete i due termini letteralmente dall'inglese vi sarà evidente la metafora del consumatore/fornitore di servizi - spesso si fa' riferimento ai ruoli nel ristorante di cliente/cameriere]. Il server è un programma che raccoglie le richieste di un client e le esegue. In tale sistema il client è responsabile delle operazioni, mentre il server non prende alcuna iniziativa da solo.

### 9.2.1 Secure Remote Login: SSH

Il protocollo SSH (Secured SHell) è stato designato per garantire sicurezza ed affidabilità. Le connessioni che utilizzano il protocollo SSH sono sicure: il membro associato al servizio è autenticato e tutti i dati interscambiati sono crittografati.

#### DIZIONARIO

Autenticazione e criptazione

Se avrete necessità di concedere ad un client la possibilità di eseguire o attivare azioni su un server, la sicurezza per voi giocherà un ruolo fondamentale. Dovrete quindi verificare l'identità del client e ciò sarà possibile solo attraverso un'autenticazione. Questa identità spesso altro non è che una password, che deve essere mantenuta segreta onde evitare che un altro client sia in grado di recuperarla. Ecco la necessità della criptazione, un modello crittografico che consente a due sistemi di comunicare informazioni riservate su un canale pubblico e di renderle illeggibili se intercettate da terze parti. L'autenticazione e la criptazione sono spesso argomentate insieme, in quanto entrambe generalmente coinvolte per il loro stesso impiego e perché messe in atto grazie a concetti matematici simili [Un concetto matematico è un'idea generale alla base di un'equazione, una formula, ecc.]

**CULTURA**  
Telnet e RSH  
sono ormai  
obsolete

Prima dell'avvento di SSH, Telnet e RSH erano i tools più utilizzati per l'accesso da remoto. Ormai sono obsoleti e non dovrebbero più essere utilizzati (nonostante Debian continui a supportarli).

SSH inoltre supporta due File Transfer Services. `scp` è un command line tool che funziona similmente a `cp`, tranne per il fatto che qualsiasi path [percorso] verso/di un'altra macchina deve essere preceduto dal nome della macchina seguito da due punti.

```
$ scp file machine:/tmp/
```

`sftp` è un comando interattivo molto simile a `ftp`. Pertanto, la stessa sessione `sftp` può trasferire diversi files oppure gestire i files remoti (eliminarli, cambiare il loro nome o i loro permessi, ecc.). Debian utilizza OpenSSH, una versione gratuita di SSH manutenuta dal progetto OpenBSD (un sistema operativo gratuito basato su kernel BSD e che si concentra sulla sicurezza) e fork del software originale di SSH sviluppato dalla società finlandese SSH Communications Security. Questa compagnia inizialmente sviluppava SSH come free software, ma poi ha deciso di continuare con una licenza con diritti di proprietà intellettuale. Il progetto OpenBSD ha di conseguenza creato OpenSSH per mantenere una versione gratuita di SSH.

**BASILARE**  
Fork

Un fork in ambito software indica un nuovo progetto, inizialmente clone di un altro progetto esistente [da cui è stato ispirato] e la cui intenzione è di migliorare il progetto originale. Questi due software nati identici sono destinati a divergere rapidamente in termini di sviluppo. Un fork spesso è il frutto di un disaccordo nel development team del progetto originale.

La possibilità di realizzare un fork deriva direttamente dalla natura stessa del free software; un fork ha davvero senso se consente la creazione di un'ulteriore derivata sotto licenza free software (qualora decida di cambiare il tipo di licenza rispetto a quella originale). Un fork nato da un disaccordo tecnico o relazionale altri non è che uno spreco di risorse umane; è preferibile risolvere piuttosto i disaccordi. Difatti non è raro che due progetti, di cui uno fork dell'altro, decidano poi di ricongiungersi nuovamente fondendosi.

OpenSSH è suddiviso in due pacchetti. Il lato client si trova nel pacchetto `openssh-client`, il lato server si trova in `openssh-server`. Il metapacchetto `ssh` dipende da entrambi i due pacchetti e facilita la loro installazione congiunta (`apt install ssh`).

### 9.2.1.1 Autenticazione basata su chiave

Un server remoto richiede la password ad ogni tentativo di connessione tramite SSH. Ciò potrebbe costituire un problema se desiderate automatizzare una connessione o se utilizzate un tool che richiede frequenti connessioni SSH. Per questo motivo SSH supporta un sistema di autenticazione basata su chiave.

In pratica ...

L'utente genera una coppia di chiavi sulla macchina client attraverso `ssh-keygen -t rsa`; la chiave pubblica viene memorizzata in `~/.ssh/id_rsa.pub` mentre la corrispondente chiave privata viene conservata in `~/.ssh/id_rsa`.

L'utente utilizza di conseguenza server `ssh-copy-id` per aggiungere la propria chiave pubblica al file `~/.ssh/authorized_keys` del server. Se la chiave privata non è stata protetta con una passphrase al momento della sua creazione, tutte le susseguenti connessioni al server si stabiliranno senza l'ausilio di una password. Diversamente, l'utente dovrà decodificare la chiave privata ad ogni connessione attraverso l'immissione di una passphrase. Fortunatamente `ssh-agent` consente all'utente di mantenere in memoria la chiave privata in modo che non debba reinserire regolarmente la password. Per fare ciò, l'utente dovrà solo utilizzare `ssh-add` (ad ogni sessione di lavoro) a condizione che `ssh-agent` sia già in esecuzione nella summenzionata sessione. Debian attiva `ssh-agent` per impostazione predefinita nella sessione grafica; `ssh-agent` può essere disabilitato nella sessione grafica modificando `/etc/X11/Xsession.options`. Durante le sessioni da console, l'utente dovrà avviare `ssh-agent` manualmente attraverso eval \$(`ssh-agent`).

#### CULTURA

La falla di OpenSSL in Debian Etch [Debian GNU/Linux 4.0 è stato rilasciato l'8 Aprile 2007]

La libreria OpenSSL supportata originariamente in Debian Etch presentava delle serie problematiche con il suo Random Number Generator (RNG). Il manutentore Debian aveva difatti apportato una modifica in modo che le applicazioni che facevano uso della summenzionata libreria non generassero avvisi se analizzati da memory testing tool come ad esempio valgrind. Sfortunatamente, questo cambiamento ebbe anche come conseguenza che l'RNG impiegava soltanto una sorgente di entropia corrispondente ad un process number (PID), con circa 32.000 valori possibili che non garantivano sufficiente casualità. [Molto genericamente l'entropia è un ordine di grandezza associato alla casualità o al disordine].

♦ <https://www.debian.org/security/2008/dsa-1571>

Concretamente, ogni volta che OpenSSL veniva usato per generare una chiave, produceva sistematicamente una chiave entro una serie nota di alcune centinaia di migliaia di chiavi (32.000, moltiplicato per un piccolo numero di key-lengths). [Key-length o key size è il numero di bits adoperati da un algoritmo per criptare o decriptare una chiave].

Questa anomalia afflisse chiavi SSH, chiavi SSL e certificati X.509 utilizzati da molte applicazioni come OpenVPN. Un cracker doveva pertanto solo testare tutte le chiavi [note] per cercare di ottenere un accesso non autorizzato. Per ridurre l'impatto del problema, fu modificato il demone SSH in modo che rifiutasse le chiavi problematiche elencate nei pacchetti `openssh-blacklist` e `openssh-blacklist-extra`. Inoltre, il programma `ssh-vulnkey` consente l'identificazione delle potenziali chiavi compromesse presenti nel sistema.

Un'analisi più approfondita del suddetto incidente chiarisce che è il frutto di molteplici (piccole) problematiche sia in termini di progetto OpenSSL che di manutenzione del pacchetto Debian. Una libreria ampiamente usata come OpenSSL non dovrebbe - senza modifiche - generare avvisi se testata da valgrind. Inoltre, il codice (in particolare nelle parti più sensibili come l'RNG) dovrebbe essere oggetto di più attenzioni e critiche onde evitare tali errori. L'intenzione del manutentore Debian era di convalidare le modifiche attraverso gli stessi sviluppatori OpenSSL, distribuendo le modifiche senza un'opportuna patch da revisionare e trascurando i suoi doveri derivanti dal suo ruolo all'interno di Debian. Infine, le sue scelte di manutenzione non erano ottimali: le modifiche apportate al software originale non sono state documentate in modo chiaro; tutte le modifiche sono state di fatto memorizzate in un repository "Subversion" e poi ammucchiate in una singola patch durante la creazione del pacchetto sorgente. In queste condizioni è difficile trovare le "corrective measures" [espressione derivata dal common law, può essere tradotta non letteralmente "le corrette contromisure di sicurezza"] per scongiurare che tali incidenti si ripetano. La lezione che dovrete imparare dal summenzionato incidente è che ogni divergenza, introdotta da Debian, al software upstream [a monte] deve essere giustificata, documentata, sottoposta alla supervisione del progetto upstream stesso (se possibile) ed ampiamente pubblicizzata. È in quest'ottica che sono stati sviluppati il nuovo formato del pacchetto sorgente ("3.0(quilt)") e lo `Debian sources webservice`.

♦ <https://sources.debian.org>

<b>SICUREZZA</b> Protezione della chiave privata	Chiunque possieda la chiave privata può accedere al relativo account configurato. Per questo motivo l'accesso alla chiave privata è protetto da una "passphrase". In questo modo se qualcuno recupera una copia del file contenente la chiave privata (ad esempio <code>~/.ssh/id_rsa</code> ) dovrà trovare anche la passphrase per poterla utilizzare. Questa protezione aggiuntiva non è tuttavia esente da violazioni e se sospettate che questo file sia stato compromesso, disattivate la correlata chiave sui computers in cui è stata installata rimuovendola dai files <code>authorized_keys</code> e sostituendola generando una nuova chiave.
-----------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 9.2.1.2 Come da usare da remoto le applicazioni X11

Il protocollo SSH consente di inoltrare i graphical data (denominati sessione "X11", dal nome del sistema grafico più comune di Unix); il server peraltro riserva loro un canale dati dedicato. In pratica, un'applicazione grafica eseguita da remoto può essere visualizzata attraverso il server X.org dello schermo locale e l'intera sessione (input e display) sarà sicura. Dal momento che questa funzionalità consente alle applicazioni da remoto di interferire con il sistema locale, è disabilitata per impostazione predefinita. Per poterla attivare dovete specificare `X11Forwarding yes` nel file di configurazione `/etc/ssh/sshd_config` del server. Inoltre l'utente dovrà richiedere esplicitamente tale funzionalità aggiungendo l'opzione `-X` all' ssh command-line.

### 9.2.1.3 Port Forwarding: come creare gli Encrypted Tunnels

Le opzioni `-R` ed `-L` consentono a ssh di creare "tunnel crittografati" tra due macchine, inoltrando in modo sicuro un local TCP port (andate a leggere la casella di testo "TCP/UDP" a pagina 238) ad una macchina remota o viceversa.

<b>DIZIONARIO</b> Tunnel	La rete Internet e la maggior parte delle LAN's - Local Area Networks [reti locali] ad essa collegate funzionano in packetmode e non in connected mode, ciò significa che un pacchetto inviato da un computer all'altro si fermerà su diversi routers intermediari per essere così instradato verso la propria destinazione. In ogni caso potrete comunque simulare una connected operation [genericamente una "connessione diretta"] in cui il flusso dati viene encapsulato in normali pacchetti IP. Questi pacchetti seguiranno il loro solito percorso, ma il flusso dati sarà ricostruito invariato alla destinazione. Il percorso viene definito "tunnel", per l'analogia con un tunnel stradale, in cui i veicoli guidano direttamente dall'entrata (input) all'uscita (output) senza incontrare incroci, diversamente da un percorso stradale in superficie che potrebbe presentare incroci e cambi di direzione. Per di più potrete sfruttare la summenzionata operazione per crittografare il tunnel: così facendo il flusso dati, che scorrerà attraverso il tunnel, sarà irriconoscibile dall'esterno, ma ritornerà in chiaro e decriptato all'uscita del tunnel.
-----------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

`ssh -L 8000:server:25` stabilisce una sessione SSH con l'host `intermediary` e si mette in ricezione del local port 8000 (andate a vedere la Figura 9.3 "Come viene inoltrato un local port tramite SSH" a pag. 211). Qualsiasi connessione stabilita su questo port farà sì che ssh avvii a sua volta una connessione dal computer `intermediary` al port 25 sul server, intercomettendo entrambe le connessioni.

`ssh -R 8000:server:25` stabilisce una sessione SSH con il computer `intermediary`, ma in questo caso ssh si mette in ricezione del local port 8000 dello stesso computer `intermediary` (andate a vedere la Figura 9.4 "Come viene inoltrato un remote port tramite SSH" a pag. 211). Pertanto qualsiasi connessione stabilita su questo port determinerà una connessione ssh dalla macchina local al port 25 sul server, interconnettendo entrambe le connessioni.

In entrambi i casi, sono state stabilite delle connessioni al port 25 sull'host server, che attraversano il tunnel SSH stabilito tra la macchina local e la macchina intermediary. Nel primo caso, l'ingresso del tunnel è il local port 8000 e i dati viaggiano verso intermediary prima di andare al server sulla rete "pubblica". Nel secondo caso, l'input e l'output del tunnel sono invertiti: l'ingresso è il port 8000 sulla macchina intermediary, mentre l'output è l'host local che invia i dati al server. In pratica generalmente il server è la macchina local o intermediary. In questo modo SSH protegge la connessione da un capo all'altro.

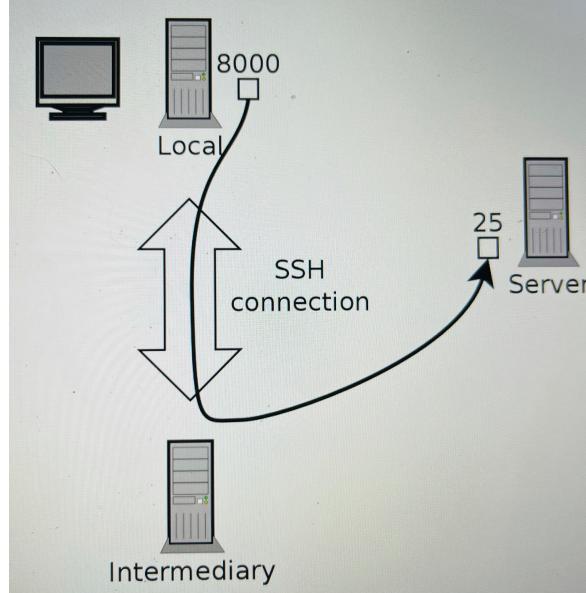


Figura 9.3 Come viene inoltrato [Forwarding] un local port tramite SSH

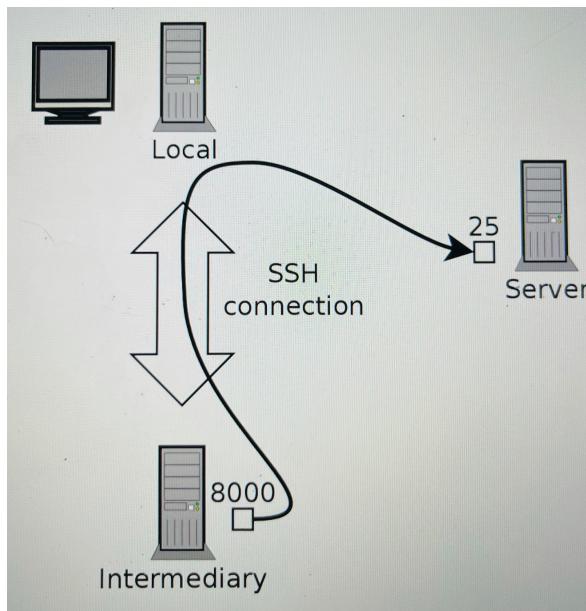


Figura 9.4 Come viene inoltrato [Forwarding] un remote port tramite SSH

### 9.2.2 Come utilizzare i Remote Graphical Desktops

Il VNC (Virtual Network Computing) consente l'accesso da remoto ai graphical desktops.

Questo tool viene utilizzato principalmente per l'assistenza tecnica: l'amministratore può visualizzare gli errori che riscontra l'utente e mostrargli la corretta sequenza di controffensive senza dovergli stare accanto.

L'utente deve innanzitutto autorizzare la condivisione della propria sessione. GNOME, un ambiente desktop con interfaccia grafica, include questa opzione a partire da Jessie nel suo pannello di controllo (contrariamente alle versioni precedenti in cui era necessario installare ed eseguire vino). KDE Plasma, al momento della stesura di questo testo, richiede krfb per poter condividere la sessione corrente tramite VNC. Il comando x11vnc (dell'omonimo pacchetto Debian) negli altri ambienti desktop con interfaccia grafica offre lo stesso servizio; potrete renderlo accessibile all'utente tramite un'icona esplicita.

Se la sessione grafica avviene tramite VNC, l'amministratore deve connettersi attraverso un VNC client. GNOME usa vinagre e remmina per consentire tale servizio, mentre il progetto KDE include krdc (attraverso il menu K -> Internet -> Remote Desktop Client). Ci sono anche altri VNC client utilizzabili da riga di comando, tra cui xvnc4viewer, dall'omonimo pacchetto Debian. Una volta connesso, l'amministratore potrà esaminare cosa sta succedendo e persino intervenire sulla macchina da remoto, mostrando all'utente come procedere.

#### SICUREZZA VNC tramite SSH

Se desiderate connettervi tramite VNC in modo che i dati non siano trasmessi in cleartext sulla rete [genericamente "in cleartext" si riferisce alla modalità di trasmissione dei dati in chiaro ovvero non criptati], è possibile incapsularli in un tunnel SSH (andate a leggere il paragrafo 9.2.1.3 "Port Forwarding: come creare gli Encrypted Tunnels" a pag. 210). Dovete però sapere che VNC utilizza per impostazione predefinita il port 5900 per il primo screen (denominato "localhost:0"), 5901 per il secondo (denominato "localhost:1") e così via. Il comando ssh -L localhost:5901:localhost:5900 -N -T machine crea un tunnel tra il local port 5901 dell'interfaccia localhost ed il port 5900 del computer machine. Il primo "localhost" costringe SSH a mettersi in ricezione solo di quell'interfaccia del computer locale. Il secondo "localhost" indica l'interfaccia del computer machine (in remoto) che riceverà il traffico di rete in ingresso di localhost:5901. Di conseguenza vncviewer localhost:1 collegherà il VNC client al remote screen nonostante abbiate indicato il nome della macchina locale. Una volta terminata la sessione VNC, ricordatevi di chiudere sia il tunnel, sia la correlata sessione SSH.

#### BASILARE Display Manager

gdm3, kdm, lightdm e xdm sono Display Managers. Prendono il controllo dell'interfaccia grafica poco dopo il boot per offrire all'utente una login screen. Una volta autenticato l'utente, il display manager esegue i programmi necessari per avviare una sessione di lavoro in modalità grafica.

VNC è utile anche per i mobile users oppure per i dirigenti di azienda che per esigenze occasionali si connettono da casa per ottenere un accesso attraverso un desktop remoto simile a quello che hanno al lavoro. La configurazione di un servizio di tale portata è più complicata: dovete prima installare il pacchetto vnc4server, modificare la configurazione del display manager in modo che le richieste di XDMCP query (con gdm3 vi basterà aggiungere Enable=true nella sezione "xdmcp" di /etc/gdm3/daemon.conf) ed infine avviare il VNC server tramite inetd in modo che la sessione VNC venga avviata non appena un utente tenta di loggarsi. Per esempio potreste aggiungere questa riga in /etc/inetd.conf:

```
5950 stream tcp nowait nobody.tty /usr/bin/Xvnc Xvnc -inetd -query localhost -> once -geometry 1024x768 -depth 16 securitytypes=none
```

Il reindirizzamento delle connessioni in entrata al display manager risolve il problema dell'autenticazione, perché solo gli utenti con account locali saranno in grado di oltrepassare la gdm3 login screen (o l'equivalente kdm, xdm, ecc.). Dal momento che questa operazione consente tranquillamente accessi simultanei multipli (a patto che il server sia abbastanza potente), può anche essere utilizzata per offrire desktop completi ai mobile users (oppure a sistemi desktop meno potenti, configurati come thin clients). Gli utenti effettuano il login semplicemente al server's screen attraverso vncviewer server:50, in quanto il port utilizzato è il 5950.

### 9.3. Gestione dei diritti [permessi]

Linux è un sistema multiutente dichiarato, di conseguenza è stato doverosamente equipaggiato di un sistema di gestione dei permessi per controllare l'insieme delle operazioni autorizzate su files e directories, che include tutte le risorse ed i dispositivi di sistema (su un sistema Unix, qualsiasi dispositivo è rappresentato da un file o da una directory). Questo principio è comune a tutti i sistemi Unix, ma un vademecum può rivelarsi sempre utile, soprattutto se ci sono alcuni usi avanzati meritevoli di attenzione e relativamente sconosciuti.

Ogni file o directory possiede dei permessi tipici per tre categorie di utenti:

- owner [utente-titolare dei diritti] (rappresentato dalla lettera "u", la prima lettera del lemma inglese "user");
- owner group [gruppo-titolare dei diritti] (rappresentato dalla lettera "g", la prima lettera del lemma inglese "group"), si riferisce e sono inclusi tutti i membri del gruppo;
- others [accessibile anche per tutti gli altri utenti] (rappresentato dalla lettera "o", la prima lettera del lemma inglese "other").

I sopra citati tre tipi di diritti possono essere combinati con i sottostanti permessi:

- reading [lettura] (rappresentato dalla lettera "r", la prima lettera del lemma inglese "read");
- writing [scrittura o modifica] (rappresentato dalla lettera "w", la prima lettera del lemma inglese "write");
- executing [esecuzione] (rappresentato dalla lettera "x", la seconda lettera del lemma inglese "execute").

Riferiti ad un file, i permessi si sottintendono facilmente: il "permesso di lettura" consente la lettura del contenuto (inclusa la copia); il "permesso di scrittura" consente di modificarlo ed il "permesso di esecuzione" consente di eseguirlo (solo nel caso in cui il file è un programma).

#### SICUREZZA

Eseguibili: setuid e setgid

Esistono due diritti specifici per i files eseguibili: **setuid** e **setgid** (rappresentati dalla lettera "s"). Occorre precisare che si tratta spesso di "bit" in quanto ciascuno di questi valori booleani è rappresentato individualmente da uno 0 o da un 1. Questi due diritti consentono a qualsiasi utente di eseguire il programma in questione rispettivamente con i diritti owner [per setuid] o con i diritti owner group [per setgid]. Questo meccanismo consente l'accesso a funzionalità che richiedono permessi di più alto livello rispetto a quelli che normalmente dovrebbero essere disponibili.  
Dal momento che un setuid root program viene eseguito sistematicamente sotto l'identità del super-user [con conseguenti diritti e permessi], è molto importante verificarne la sicurezza e l'affidabilità. Difatti un utente malintenzionato potrebbe manometterlo affinché chiama un comando a sua scelta allo scopo di assumere l'identità di root e di acquisire tutti i diritti sul sistema.

Una directory viene gestita in modo diverso: il "permesso di lettura" conferisce il diritto di consultare l'elenco delle sue voci (files e directories); il "permesso di scrittura" conferisce il diritto di

creare o eliminare files; il "permesso di esecuzione" conferisce il diritto di poter navigare nella cartella (per giungere alla cartella dovete utilizzare nello specifico il comando cd). Il permesso di navigare in una directory senza il permesso di poterla leggerla conferisce il permesso di accedere solo a quei contenuti dei quali se ne conosce l'esistenza nonché il nome esatto, altrimenti saranno irreperibili.

#### SICUREZZA

##### setgid directory e sticky bit

Il **setgid** bit si applica anche alle directories. A tutte le voci create all'interno delle directories verrà quindi assegnato l'**owner group** della parent directory, anziché ereditare il main group del creatore come di consueto. Ciò impedirà all'utente di modificare il main group (utilizzando il comando newgrp) quando lavora in un file tree condiviso con altri utenti dello stesso gruppo dedicato.

Lo **sticky bit** (rappresentato dalla lettera "t") è un permesso che è utile solo nelle directories. Viene utilizzato in particolare per le directories temporanee in cui chiunque ha il "permesso di scrittura" (come ad esempio /tmp/): limita l'autorizzazione alla cancellazione dei files solo a coloro i quali ne sono titolari di diritti o che fanno parte dell'owner group della parent directory. In assenza dello **sticky bit**, chiunque potrebbe cancellare i files di altri utenti in /tmp/.

Questi tre comandi gestiscono i permessi associati ad un file:

- **chown user file** cambia l'owner di un file;
- **chgrp group file** altera l'owner group;
- **chmod rights file** modifica i permessi del file.

Esistono due metodi per rappresentare il titolo relativo ai diritti. La rappresentazione simbolica, fra i due metodi, è senza dubbio la più facile da comprendere e memorizzare. È stata precedentemente trattata quando sono state citate le "lettere simboliche" che rappresentano le categorie degli utenti, nonché i permessi. Potrete definire i diritti per ogni categoria di utenti (u/g/o), impostandoli esplicitamente (=), aggiungendoli (+) o sottraendoli (-). La formula u=rwx, g+rw, o-r di conseguenza: conferisce all'owner i permessi di lettura, scrittura ed esecuzione; aggiunge all'owner group i permessi di lettura e scrittura; rimuove agli others il permesso di lettura. I diritti non coinvolti dalle operazioni di addizione o sottrazione rimangono invariati. La lettera "a" sta per "all" e copre tutte e tre le categorie di utenti, pertanto a=rx attribuisce a tutte e tre le categorie gli stessi permessi (lettura ed esecuzione, ma non di scrittura).

La rappresentazione numerica (ottale) associa ciascun permesso ad un valore numerico: 4 per la lettura, 2 per la scrittura e 1 per l'esecuzione. Quindi dovete associare ciascuna combinazione di diritti con la somma di queste cifre. Ciascun valore viene poi assegnato alle diverse categorie di utenti posizionandolo in un ordine prefissato (owner, group, others).

Per esempio il comando chmod 754 file imposta i seguenti diritti: lettura, scrittura ed esecuzione all'owner (in quanto  $7 = 4 + 2 + 1$ ); lettura ed esecuzione al group (in quanto  $5 = 4 + 1$ ); solo lettura agli others. Il numero 0 significa nessuna autorizzazione, quindi il comando chmod 600 file consente all'owner solo permessi di lettura/scrittura, non conferendo alcun permesso alle altre due categorie di utenti restanti. I diritti più frequenti sono 755 per i files eseguibili o le directory e 644 per i data files.

Per rappresentare i diritti speciali, potrete aggiungere alle tre cifre una quarta cifra seguendo lo stesso principio di prima, premettendo che setuid, setgid e sticky bit sono associati rispettivamente ai valori numerici 4, 2 e 1. chmod 4754 assocerà quindi il setuid bit ai permessi rappresentati dalle precedenti cifre numeriche.

Occorre precisare che l'uso della notazione numerica ottale consente solo di modificare tutti i diritti su un file in un'unica soluzione; non potrete usarla semplicemente per aggiungere, ad esempio, i permessi di lettura per l'owner group, in quanto dovete anche tenere conto dei permessi/diritti pregressi e calcolare il nuovo valore numerico corrispondente.

#### **SUGGERIMENTO** Metodo ricorsivo

A volte è necessario modificare le autorizzazioni di un intero file tree. Tutti i comandi appena descritti hanno quindi un'opzione `-R`, che esegue ricorsivamente l'operazione richiesta nelle sub-directories. La distinzione tra directories e files è spesso causa di problematiche durante le operazioni ricorsive. Per questo motivo è stata introdotta la lettera "x" nella rappresentazione simbolica dei diritti. Rappresenta il permesso di esecuzione e sarà applicato solo alle directories (ma non ai files che non hanno ancora questo diritto). Pertanto, `chmod -R a+X directory` aggiungerà i permessi di esecuzione a tutte le categorie di utenti (a) su tutte le subdirectories e per quel che concerne i files, solo su quei files su cui almeno una categoria di utenti (anche un unico owner) abbia già i permessi di esecuzione.

#### **SUGGERIMENTO** Come cambiare l'user ed il group

Spesso si desidera cambiare il group e l'owner di un file contemporaneamente. Il comando `chown` supporta una sintassi speciale per farlo: `chown user:group file`

#### **ANDANDO OLTRE** umask

Quando un'applicazione crea un file, gli assegna dei permessi indicativi, con la consapevolezza che il sistema revoca automaticamente determinati permessi, in base alle istruzioni ricevute dal comando `umask`. Difatti immettendo in una shell il comando `umask` visualizzerete una mask, ad esempio `0022`. Si tratta di una rappresentazione ottale dei permessi che devono essere sistematicamente rimossi (in questo caso, i permessi di scrittura per il gruppo ed altri utenti). Se trasmetterete un nuovo valore ottale, il comando `umask` vi consentirà di cambiare la mask. Utilizzato in un `shell initialization file` (ad esempio `~/.bash_profile`), cambierà la maschera predefinita per le vostre sessioni di lavoro.

## **9.4. Interfacce di amministrazione**

L'impiego di un'interfaccia grafica per "amministrare" è valevole in diverse circostanze. Un amministratore potrebbe non necessariamente conoscere tutti i dettagli per la configurazione dei servizi e potrebbe non avere il tempo di cercare la documentazione inherente. Un'interfaccia di amministrazione grafica accelererà di fatto il deployment di un nuovo servizio. [Per deployment si intendono tutte quelle attività che rendono funzionale un sistema operativo quali ad esempio l'installazione, la configurazione comprensiva di eventuale adattamento del sistema in base ai servizi che i suoi utenti devono svolgere, l'addestramento del personale, la verifica, l'aggiornamento, la sincronizzazione, ecc.]

Inoltre, può semplificare il settaggio dei servizi più ostici da configurare.

Questo di tipo interfaccia rappresenta solo una facilitazione e non è definitiva. Difatti in ogni caso l'amministratore dovrà padroneggiarne il funzionamento in modo da comprenderne ed aggirarne qualsiasi potenziale anomalia.

Dal momento che nessuna interfaccia è perfetta, potreste essere tentati di testare diverse soluzioni. Vi consigliamo di evitarlo, in quanto i tools fra loro potrebbero avere un funzionamento incompatibile. Inoltre sebbene tutti i tools mirino ad una grande flessibilità e tentino di adottare il file di configurazione come unico riferimento, non sono sempre in grado di integrare modifiche esterne.

### **9.4.1. Come amministrare attraverso un'interfaccia web: webmin**

È probabilmente una delle interfacce di amministrazione più riuscite. È un sistema modulare gestibile attraverso un browser Web, che si occupa di una vasta serie di aree e di tools. Inoltre, è internazionalizzato e disponibile in molte lingue.

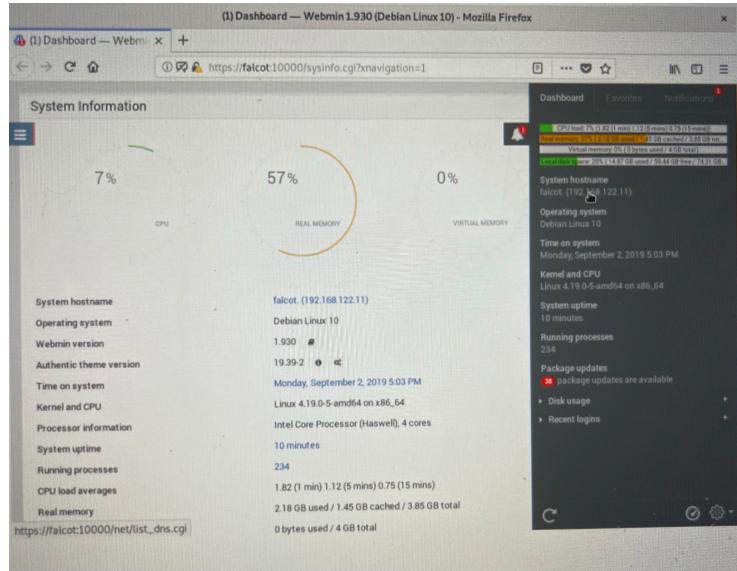


Figura 9.5 Dashboard di Webmin

Sfortunatamente, webmin non è più disponibile ufficialmente su Debian. Il Debian maintainer – Jaldhar H. Vyas – l'ha rimosso non avendo tempo per manutenerlo ad un livello di qualità accettabile. Dato che nessuno è stato ufficialmente incaricato di sostituire il sopramenzionato maintainer, Buster non ha più alcun pacchetto webmin.

Esiste, tuttavia, un pacchetto non ufficiale, distribuito sul sito [webmin.com](http://webmin.com). A differenza dei pacchetti in passato disponibili su Debian, quest'ultimo è monolitico: tutti i moduli di configurazione sono installati e attivati di default anche se il servizio corrispondente non è installato sulla macchina.

#### SICUREZZA

Cambiate la password di root

Durante il primo login, l'identificazione viene eseguita con `root` come `username` e la vostra `root` `password`. Tuttavia, vi consigliamo di cambiare la `password` utilizzata per `webmin` il prima possibile; così, qualora fosse stata compromessa, nonostante conceda importanti diritti amministrativi sulla macchina, la `root` `password` del server non sarà coinvolta.

Fate Attenzione! `webmin` ha diverse funzioni ed un utente malintenzionato, in grado di ottenerne l'accesso, potrebbe potenzialmente compromettere la sicurezza dell'intero sistema. In generale, interfacce di questo tipo non sono consigliate su sistemi di grandi dimensioni con considerevoli vincoli di sicurezza (firewall, sensitive servers, ecc.).

`webmin` può essere utilizzato tramite un'interfaccia web, ma non richiede necessariamente l'installazione di Apache. Difatti questo software ha un proprio mini web server integrato. Quest'ultimo è in ricezione per impostazione predefinita del port 10.000 e accetta connessioni HTTP sicure.

I moduli integrati sono dedicati ad una vasta varietà di servizi, tra cui:

- tutti i servizi basilari – creazione utenti e gruppi; gestione dei crontab files, init scripts; consultazione dei logs, ecc;
- bind: configurazione del server DNS (name service);
- postfix: configurazione del server SMTP (email);
- inetd: configurazione dell/inetd super-server;

- **quota**: user quota management;
- **dhcpd**: configurazione del server DHCP;
- **proftpd**: configurazione del server FTP;
- **samba**: configurazione del Samba file server;
- **software**: installazione o rimozione di software dei pacchetti Debian ed aggiornamento di sistema.

L'interfaccia di amministrazione è accessibile da un browser Web all'indirizzo <https://localhost:10000>. Fate attenzione! Non tutti i moduli sono direttamente utilizzabili. Qualche volta è necessario configurarli specificando le posizioni dei correlati files di configurazione e di files eseguibili (programmi). Se il sistema non sarà in grado di far funzionare un modulo necessario vi interpellerà appropriatamente.

#### **ALTERNATIVA GNOME control center**

Il progetto GNOME offre anche diverse interfacce di amministrazione, generalmente accessibili tramite la voce "Settings" nel menu-utente nella parte superiore destra dello schermo. `gnome-control-center` è il programma principale che li riunisce, ma molti vasti tools di configurazione del sistema sono di fatto supportati da altri pacchetti (`accounts-service`, `system-config-printer`, ecc.). Anche se queste applicazioni sono facili da usare, gestiscono solo una piccola parte dei servizi-base: gestione utenti; configurazione dell'orologio; configurazione della rete; configurazione delle stampanti; ecc.

#### **9.4.2. Configurazione dei pacchetti: debconf**

Diversi pacchetti vengono configurati automaticamente dopo alcune domande durante l'installazione, richieste dal Debconf tool. Questi pacchetti possono essere riconfigurati eseguendo `dpkg-reconfigure package`.

Nella maggior parte dei casi, queste impostazioni sono molto semplici: vengono modificate solo alcune variabili importanti nel file di configurazione. Queste variabili sono talvolta messe assieme tra due linee di "demarcazione" in modo che una riconfigurazione del pacchetto limiti la sua influenza all'area che esse delimitano. In altri casi, la riconfigurazione non effettuerà cambiamenti se lo script rileva una modifica manuale del file di configurazione, al fine di preservare l'operato "umano" (in quanto lo script non può garantire che le proprie modifiche non disgregheranno le impostazioni preesistenti).

#### **DEBIAN POLICY Preserva i cambiamenti**

La Debian Policy sancisce esplicitamente che deve essere fatto di tutto per preservare le modifiche manuali apportate ai files di configurazione, di conseguenza sempre più scripts prendono le dovute precauzioni quando modificano i files di configurazione. Il principio generale è semplice: lo script apporta delle modifiche solo se conosce lo status del file di configurazione, che viene verificato confrontando il checksum del file con quello dell'ultimo file generato automaticamente. Se corrispondono, lo script è autorizzato a modificare il file di configurazione. In caso contrario, ritenendo che sia stato modificato manualmente, richiede quale azione debba intraprendere (install the new file [installa il nuovo file], save the old file [salva il vecchio file], try to integrate the new changes with the existing file [tenta ad integrare le nuove modifiche nel file preesistente]). Il summenzionato principio di precauzione è stato adottato da Debian ed ignorato dalle altre distribuzioni per lungo tempo, ma adesso anche le seconde lo stanno abbracciando. Il programma `ucf` (dell'omonimo pacchetto Debian) può essere utilizzato per implementare tale funzionalità.

## **9.5. Eventi di sistema di syslog**

### **9.5.1. Principi e funzionamento**

Il demone `rsyslogd` è responsabile della raccolta dei messaggi di servizio provenienti dalle applicazioni e dal kernel della loro distribuzione nei log files (generalmente conservati nella directory `/var/log/`). Risponde al file di configurazione `/etc/rsyslog.conf`.

Ciascun log file è associato ad un application subsystem (denominato "facility" nella documentazione):

- `auth` e `authpriv`: per l'autenticazione;
- `cron`: proviene dai task scheduling service, `cron` e `atd`;
- `daemon`: riguarda i demoni senza una particolare classificazione (DNS, NTP, ecc.);
- `ftp`: riguarda server;
- `kern`: messaggio proveniente dal kernel;
- `lpr`: proviene dal printing subsystem;
- `mail`: proviene dall'email subsystem;
- `news`: messaggio dell'Usenet subsystem (in particolare dal NNTP server - Network News Transfer Protocol);
- `syslog`: messaggio proveniente dallo stesso `syslogd` server;
- `user`: messaggio user (generico);
- `uucp`: messaggio dal UUCP server (Unix to Unix Copy Program, un vecchio protocollo usato principalmente per distribuire le email);
- `local0` a `local7`: riservato per usi locali.

Ciascun messaggio è inoltre associato ad un livello di priorità. In basso l'elenco in ordine decrescente.

- `emerg`: "Aiuto!" Il sistema è probabilmente inutilizzabile;
- `alert`: sbrigatevi, perdere ancora tempo potrebbe essere pericoloso, le controffensive devono essere prese immediatamente;
- `crit`: le condizioni sono critiche;
- `err`: errore;
- `warn`: avviso (potenziale errore);
- `notice`: condizioni normali, ma il messaggio è significativo;
- `info`: messaggio informativo;
- `debug`: messaggio di debug.

### 9.5.2. Il file di configurazione

La sintassi complessa del file `/etc/rsyslog.conf` viene trattata sia nella man page `rsyslog.conf(5)`, sia nella documentazione HTML disponibile nel pacchetto `rsyslog-doc` (`/usr/share/doc/rsyslog-doc/html/index.html`). Il principio generale è scrivere delle coppie: "selector" ed "action". Il selector definisce tutti messaggi rilevanti e l'action descrive come gestirli.

#### 9.5.2.1. Sintassi del Selector

Il selector è un elenco che usa come separatore il punto e virgola e che è composto da coppie di `subsystem.priority` (ad esempio: `auth.notice; mail.info`). L'asterisco può essere utilizzato sia per rappresentare tutti i subsystems, sia per rappresentare tutte le priorities (ad esempio `*.alert o mail.*`). È possibile riunire diversi subsystems separandoli con una virgola (ad esempio: `auth,mail.info`).

La priorità indicata garantisce anche i messaggi con priorità maggiore o uguale alla sua: `auth.alert` seleziona quindi i messaggi dell' `auth` subsystem con priorità `alert` o `emerg`. L'uso del punto esclamativo (!), selezionerà le priorità minori e non uguali: `auth.!Notice` pertanto individuerà i messaggi rilasciati da `auth` con priorità `info` o `debug`. L'uso del segno uguale (=), segnala una corrispondenza esatta all'unica priorità indicata (`auth.=notice` selezionerà quindi solo i messaggi da `auth` con priorità `notice`).

Ciascun elemento elencato nel selector `override` (trad. lett. ignora o ignora e sostituisce con) gli elementi precedenti. Sarà possibile così limitare un set o escluderne diversi elementi. Ad esempio, `kern.info;kern!err` definisce i messaggi del kernel con priorità tra `info` e `warn`. La priorità `none` si riferisce ad un "empty set" (trad. lett. un set vuoto, ovvero "nessuna priorità") e può essere utilizzata per escludere un subsystem da un set di messaggi. Pertanto, `*crit;kern.none` è riferito a tutti i messaggi con priorità maggiore o uguale a `crit` che non provengono dal kernel.

#### 9.5.2.2 Sintassi delle Actions

##### ALTERNATIVA

Il named pipe, un persistent pipe

Un named pipe è un particolare tipo di file che funziona come un pipe tradizionale (ovvero il pipe che realizzate nella command-line attraverso il simbolo "|"), tranne per il fatto che utilizza un file. Questo meccanismo ha il vantaggio di essere in grado di mettere in relazione due processi non correlati. In pratica ciò che è contenuto in un named pipe file blocca un processo in fase di scrittura sino a quando un altro processo non tenta di leggere i dati scritti. Ovvero dopo che il secondo processo legge i dati scritti dal primo processo, il primo processo può riprendere la sua esecuzione.  
Un named pipe file viene creato con il comando `mknod`.

Le diverse azioni possibili sono:

- aggiungere il messaggio ad un file (ad esempio: `/var/log/messages`);
- inviare il messaggio ad un remote syslog server (ad esempio: `@log.falcot.com`);
- inviare il messaggio ad un named pipe preesistente (ad esempio: `| /dev/xconsole`);
- inviare il messaggio ad uno o più utenti, se hanno effettuato il log (ad esempio: `root,rhertzog`);

- inviare il messaggio a tutti gli utenti che hanno effettuato il log (ad esempio: \*);
- scrivere il messaggio in una text console (ad esempio: /dev/tty8).

#### SICUREZZA

Forwarding logs  
[Inoltrare i logs]

È consigliabile registrare i logs più importanti su una macchina separata (meglio se dedicata a tale scopo), in quanto non consentirà a dei malintenzionati di rimuovere le tracce del loro operato (a meno che non abbiano anche compromesso il server dedicato). Per di più qualora capitasse una grave anomalia (come ad esempio un crash del kernel), i logs registrati su un'altra macchina dedicata vi consentiranno di ricostruire la sequenza degli eventi che hanno determinato la problematica.

Per accettare i messaggi di logs inviati da altre macchine dovete riconfigurare `rsyslog`: in pratica, è sufficiente attivare le voci già presenti e pronte all'uso in `/etc/rsyslog.conf` (`$ModLoad imudp` e `$UDPServerRun 514`).

## 9.6. L'inetd Super-Server

`inetd` (spesso definito "Internet super-server") è in realtà un server per i servers. Mette in esecuzione su richiesta i servers usati raramente, in modo che non debbano rimanere attivi continuamente.

Il file `/etc/inetd.conf` elenca i suddetti servers ed i loro ordinari ports. Il comando `inetd` si mette in ricezione di ogni cosa che appartiene a tali servers; se rileva una connessione su uno di essi, esegue il corrispondente server program.

#### DEBIAN POLICY

Registrare un server su `inetd.conf`

I pacchetti spesso desiderano inserire un nuovo server nel file `/etc/inetd.conf`, ma la Debian Policy proibisce a qualsiasi pacchetto di modificare un file di configurazione che non gli appartiene direttamente. Per questo motivo è stato creato lo script `update-inetd` (dell'omonimo pacchetto): gestisce il file di configurazione, in modo che gli altri pacchetti possono usarlo per chiedere al super-server di registrare un nuovo server nella configurazione dello stesso super-server.

Ogni riga significativa nel file `/etc/inetd.conf` descrive un servizio attraverso sette fields (separati da spazi):

- Il TCP o UDP port number, oppure il service name (che viene associato ad un numero di port standard tramite il file `/etc/services`).
- Il socket type: `stream` per una connessione TCP, `dgram` per datagrammi UDP;
- Il protocol: `tcp` o `udp`.
- Le options: sono possibili due valori, `wait` o `nowait`, per indicare ad `inetd` se deve attendere la fine del processo avviato prima di accettare un'altra connessione. Per le connessioni TCP, facilmente multiplexable generalmente potrete usare `nowait`. [La multiplazione, o in inglese multiplexing, è un metodo attraverso cui diversi segnali analogici o digitali sono combinati in un unico segnale attraverso uno shared medium]. Per i programmi che rispondono su UDP, `nowait` dovrebbe essere usato solo se il server è in grado di gestire più connessioni in parallelo. Questo campo può essere suffisso con un punto seguito dal numero massimo di connessioni consentite al minuto (il limite predefinito è 256).
- L'user-name dell'utente, l'identità attraverso cui il server verrà eseguito.

- Il full path del programma server da eseguire. [Il full path o absolute path punta alla stessa posizione in un file system, indipendentemente dalla directory di lavoro corrente. Per fare ciò, dovete includere la root directory principale.]
- Gli argomenti: l'elenco completo degli argomenti del programma, elenco che include anche il nome del programma in sé (corrispondente a `argv[0]` in C).

L'esempio seguente illustra i casi più comuni:

#### Esempio 9.1 Estratto da /etc/inetd.conf

```
talk dgram udp wait nobody.tty /usr/sbin/in.talkd in.talkd
finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd
ident stream tcp nowait nobody /usr/sbin/identd identd -i
```

Il programma `tcpd` è spesso utilizzato nel file `/etc/inetd.conf`. Consente di limitare le connessioni in entrata applicando delle regole di controllo accessi (access control rules), documentate nella man page `hosts_access(5)` e configurate nei files `/etc/hosts.allow` e `/etc/hosts.deny`. Una volta stabilito che la connessione è stata autorizzata, `tcpd` a sua volta esegue il server program effettivamente richiesto (come `in.fingerd` nel soprastante esempio). Occorre farvi notare che `tcpd` si basa sul nome con cui è stato invocato (che è il primo argomento, `argv[0]`) per identificare il programma effettivo da eseguire. Quindi non dovete iniziare l'elenco degli argomenti con `tcpd` bensì con il programma che deve essere "wrapped" [genericamente, un "wrapped" o "rivestito", è il programma che di fatto esegue i servizi, mentre il "wrapper" è il programma che "rivestendolo" lo mette in esecuzione].

#### COMUNITÀ

Wietse Venema

Wietse Venema, è un esperto in sicurezza informatica e noto programmatore, autore del programma `tcpd`. È anche il main creator di Postfix, un server di posta elettronica modulare (SMTPL - Simple Mail Transfer Protocol) progettato per essere più sicuro ed affidabile di sendmail, che nel corso del tempo è stato oggetto di diverse vulnerabilità di sicurezza.

#### ALTERNATIVA

Altri comandi inetd

Sebbene Debian installi `openbsd-inetd` di default, ci sono molte alternative, ad esempio: `inetutils-inetd`, `micro-inetd`, `rlinetd` e `xinetd`. In particolare l'ultima alternativa menzionata, concretizzazione di un super server, offre alcune interessanti possibilità. Specificatamente, consente di separare la configurazione in più files (archiviati, di conseguenza, nella directory `/etc/xinetd.d/`), in modo da semplificare la vita degli amministratori. Infine, occorre anche ricordare, non essendo meno importante, che è possibile emulare il funzionamento di `inetd` attraverso il meccanismo di socket-activation di `systemd` (consultate al riguardo il paragrafo 9.1.1, "Il sistema init `systemd`" a pagina 198).

## 9.7. Scheduling Tasks attraverso cron e atd

[Genericamente, in informatica lo scheduling è il metodo attraverso cui il lavoro viene assegnato alle risorse che lo completano.]

`cron` è il demone responsabile dell'esecuzione dei comandi programmati (scheduled) e ricorrenti (giornalieri, settimanali, ecc.); `atd` gestisce i comandi che devono essere eseguiti una volta sola, ma in un momento futuro e specifico.

In un sistema Unix, molte attività sono regolarmente programmate:

- rotating dei logs;

- l'aggiornamento del database del programma `locate`;
- back-ups;
- maintenance script [trad. lett. script di manutenzione] (come ad esempio la pulizia dei file temporanei).

Per impostazione predefinita, tutti gli utenti possono pianificare l'esecuzione delle attività. Difatti ogni utente ha il proprio `crontab`, dove può registrare i comandi programmati. Ciascun utente può modificare il proprio `crontab` eseguendo `crontab -e` (il cui contenuto è memorizzato nel file `/var/spool/cron/crontabs/user`).

#### SICUREZZA

Limitare l'accesso a cron e atd

L'accesso a cron può essere limitato attraverso la creazione di un file di autorizzazione esplicita (`whitelist`), `/etc/cron.allow`, dove verranno registrati solo gli utenti autorizzati a pianificare i comandi. Tutti gli altri utenti di conseguenza verranno automaticamente privati di questa funzionalità. Diversamente, per estromettere solo uno o due "piantagrane", scrivete il loro nome nel file di divieto esplicito (`blacklist`), `/etc/cron.deny`. Potrete procedere con un iter simile anche con atd, attraverso i files `/etc/at.allow` e `/etc/at.deny`.

L'utente root ha un `crontab` personale, ma può anche usare il file `/etc/crontab` o redigere ulteriori `crontab` nella directory `/etc/cron.d/`. Queste ultime due soluzioni hanno il vantaggio di poter specificare l'identità dell'utente con cui associare l'esecuzione del comando.

Il pacchetto cron include per impostazione predefinita i comandi programmati predefiniti che vengono eseguiti:

- una volta all'ora -> i programmi in `/etc/cron.hourly/`;
- una volta al giorno -> i programmi in `/etc/cron.daily/`;
- una volta alla settimana -> i programmi in `/etc/cron.weekly/`;
- una volta al mese -> i programmi in `/etc/cron.monthly`.

Molti pacchetti Debian fanno affidamento su questo servizio: depositando maintenance scripts in queste directories, garantiscono il funzionamento ottimale dei loro servizi.

#### 9.7.1. Formato di un file crontab

#### SUGGERIMENTO

Test shortcuts per cron

cron riconosce alcune scorciatoie, con cui rimpiazzare i primi cinque fields in una voce di crontab. Queste scorciatoie corrispondono alle opzioni di pianificazione più classiche:

- @yearly: una volta all'anno (1 gennaio alle 00:00)
- @monthly: una volta al mese (il primo del mese alle 00:00)
- @weekly: una volta alla settimana (ogni domenica alle 00:00)
- @daily: una volta al giorno (ogni giorno alle 00:00)
- @hourly: una volta ogni ora (all'inizio di ogni ora)

## SPECIAL CASE cron e l'ora legale

Messo in esecuzione su Debian, cron segue i cambiamenti d'orario (sia che si tratti del cambio dell'ora dovuto all'ora legale, sia che si tratti di un significativo cambiamento dell'ora locale) al massimo delle sue potenzialità. Pertanto, i comandi che dovrebbero essere eseguiti in un momento di fatto "inesistente" (ad esempio, le 2:30 del mattino durante il cambio dell'ora legale in primavera in Francia di fatto non esistono in quanto le 2:00 a.m. dell'ora solare diventano le 3:00 a.m. dell'ora legale) vengono posticipati (cioè poco dopo il cambio di orario, ovvero nel soprannominato esempio dopo le 3:00 a.m. dell'ora legale). Al contrario, in autunno, i comandi che per colpa del cambio dell'ora solare sarebbero stati eseguiti più volte (ovvero alle 2:30 a.m. dell'ora legale e poi anche un'ora dopo ossia alle 3:30 dell'ora solare, in quanto per il cambio d'orario le 3:00 a.m. dell'ora legale tornano ad essere le 2:00 a.m. dell'ora solare) vengono eseguiti di fatto una volta sola.

Tuttavia, fate lo stesso attenzione e verificate se l'ordine secondo cui le varie attività sono programmate ed il conseguente ritardo delle attività in esecuzione sono compatibili con i suddetti vincoli per il funzionamento stesso di cron; pertanto al bisogno organizzate una pianificazione ad hoc per le due notti dell'anno in cui avviene appunto il cambio dell'ora ed in cui possono verificarsi problematiche.

Ogni riga significativa di un crontab descrive un comando programmato attraverso i seguenti sei (o sette) fields:

- il valore per il minuto (numeri che vanno da 0 a 59);
- il valore per l'ora (da 0 a 23);
- il valore per il giorno del mese (da 1 a 31);
- il valore per il mese (da 1 a 12);
- il valore per il giorno della settimana (da 0 a 7, 1 corrisponde a lunedì, domenica è rappresentato sia attraverso lo 0, sia attraverso il 7; è anche possibile utilizzare le prime tre lettere del nome del giorno in inglese, ad esempio Sun, Mon, ecc.);
- l'user name dell'identità sotto cui il comando deve essere eseguito (nel file /etc/crontab e nei frammenti depositati in /etc/cron.d/, ma non nei crontab files personali degli utenti);
- il comando da eseguire (quando sono soddisfatte le condizioni definite dalle prime cinque colonne).

[ATTENZIONE! La sintassi di crontab e di dirvish sono differenti (andate a vedere anche pag. 228). In sintesi la sintassi di crontab:

MIN = Minute (minuto), da 0 a 59

HR = Hour (ora), da 0 a 23

DOM = Day of Month (giorno del mese), da 1 a 31

MON = Month (mese) da 1 a 12

DOW = Day of Week (giorno della settimana), da 0 a 7 (0 e 7 rappresentano entrambi la domenica)]

Tutti i dettagli sono documentati nella man page crontab(5) .

Ogni valore può essere espresso sotto forma di un elenco di valori possibili (separati da virgole). La sintassi a-b descrive l'intervallo di tutti i valori tra a e b. La sintassi a-b/c descrive un intervallo con un incremento di c (ad esempio 0-10/2 corrisponde a 0,2,4,6,8,10). Il carattere \* è un wildcard character e rappresenta tutti i valori possibili. [Genericamente un wildcard character o metacarattere o carattere jolly o wild character è un singolo carattere come un asterisco che rappresenta un'insieme di caratteri o una empty string (detta anche empty word)].

### Esempio 9.3 crontab file

```
#Format  
#min hour day mon dow  command
```

```
# Download data every night at 7:25 pm  
25 19      *      *      *      $HOME/bin/get.pl
```

```
# 8:00 am,    on weekdays  (Monday through Friday)
00 08          *          *      1-5    $HOME/bin/dosomething
```

```
# Restart the IRC proxy after each reboot
@reboot /usr/bin/dircproxy
```

---

**SUGGERIMENTO**
Eseguire un comando al boot

Per eseguire un comando una sola volta, subito dopo aver avviato il computer, è possibile utilizzare la macro @reboot (un semplice riavvio di cron non attiva un comando pianificato con @reboot). Questa macro sostituisce i primi cinque campi di una voce nel crontab file.

---

**ALTERNATIVA**
Emulare cron attraverso systemd

È possibile emulare parte delle funzionalità di cron attraverso il meccanismo timer di systemd (andate a leggere il paragrafo 9.1.1., "Il sistema init systemd" a pagina 198).

### 9.7.2. Come utilizzare il comando at

Il comando at esegue un comando in uno specifico momento in futuro. Include l'ora e la data pianificate sotto forma di command-line parameters ed il comando deve essere eseguito sul suo standard input.

Il comando verrà eseguito come se fosse stato immesso nella shell corrente. at preserva l'ambiente corrente per poter riprodurre esattamente le stesse condizioni quando eseguirà il comando. L'ora designata viene espressa attraverso le seguenti ordinarie convenzioni: 16:12 o 4:12pm rappresentano le quattro e dodici minuti del pomeriggio. La data può essere espressa sia nel formato European (Occidentale Europeo), sia in quello Western (Occidentale Inglese-American) ovvero in: DD.MM.YY (ad esempio 27.07.15 che rappresenta il 27 luglio 2015), YYYY-MM-DD (ad esempio 2015-07-27 che rappresenta il 27 luglio 2015), MM/DD/[CC]YY (ad esempio 12/25/15 o 12/25/2015 che rappresentano il 25 dicembre 2015) oppure semplicemente MMDD[CC]YY (ad esempio 122515 o 12252015 che rappresentano il 25 dicembre 2015). In assenza della data, il comando verrà eseguito non appena l'orologio raggiunge l'ora designata (nello stesso giorno oppure l'indomani se l'ora designata è già trascorsa). Potrete comunque aggiungere, per essere più esplicativi, al posto della data: "today" (oggi) oppure "tomorrow" (domani).

```
$ at 09:00 27.07.15 <<END
> echo "Don't forget to wish a Happy Birthday to Raphaël!" \
>     | mail lolando@debian.org
> END
warning: commands will be executed using /bin/sh
job 31 at Mon Jul 27 09:00:00 2015
```

Per posporre l'esecuzione dopo un determinato arco tempo temporale [nell'esempio seguente number period] a partire da un momento dato [nell'esempio seguente now] viene utilizzata un'altra sintassi: at now + number period. Potrete indicare come periodo minuti, ore, giorni o settimane. Il numero indica semplicemente il numero di unità del periodo che devono trascorrere prima dell'esecuzione del comando [ad esempio 3 settimane – settimane è il periodo, mentre 3 è il numero di unità del periodo che devono trascorrere prima dell'esecuzione del comando].

Per annullare un'attività programmata attraverso cron, è sufficiente eseguire crontab -e ed eliminare la riga corrispondente nel file crontab. Annullare un'attività programmata attraverso at, è altrettanto facile, basta eseguire il comando: atrm task-number. Il numero dell'attività viene indicato dal comando at durante la programmazione dell'attività, ma potrete rintracciarlo attraverso il comando atq, che rilascia l'elenco dei comandi attualmente programmati.

## 9.8. Scheduling Asynchronous Tasks: anacron

anacron è il demone che integra cron per i computers che non rimangono accesi tutto il tempo. Dal momento che le attività regolari sono in genere programmate nel cuore della notte, non verranno mai eseguite se il computer si trova spento in quell'arco temporale. Lo scopo di anacron è di eseguire le suddette attività tenendo conto dei periodi in cui il computer non è in funzione.

Vi informiamo però anacron svolgerà le sue funzioni principalmente qualche minuto dopo l'avvio della macchina, rallentando di fatto le prestazioni del computer. Per tale ragione le attività contenute dal file `/etc/anacrontab` vengono avviate con il comando `nice`, che riduce la loro priorità di esecuzione per limitarne l'impatto sul resto del sistema. Occorre precisare, che il formato di questo file non è lo stesso di `/etc/crontab`; se avete delle esigenze particolari con anacron, consultate la man page `anacrontab(5)`.

### BASILARE

#### Priorities e nice

I sistemi Unix (e di conseguenza Linux) sono sistemi multi-tasking e multi-utente. Difatti numerosi processi possono essere eseguiti in parallelo, pur appartenendo a diversi utenti: è il kernel ad essere responsabile della distribuzione delle risorse tra i diversi processi. E per svolgere tale attività il kernel integra la nozione di priorità [priority], in modo da poter favorire determinati processi a scapito di altri, se necessario. Se un processo può essere eseguito con una bassa priorità, potrete segnalarlo ed eseguirlo con il comando `nice program`. Il programma di conseguenza usufruirà della minore percentuale possibile della CPU condivisa ed interferirà il meno possibile con gli altri processi in esecuzione. Naturalmente, se nessun altro processo necessita di essere eseguito, il programma non verrà rallentato artificialmente.

`nice` opera con livelli di "niceness" (trad. lett "cortesia"): i livelli positivi (da 1 a 19) abbassano progressivamente la priorità di un processo, diversamente i livelli negativi (da -1 a -20) innalzano progressivamente la priorità di un processo - ma solo l'utente root può usare i livelli negativi. Salvo diversa indicazione (andate a leggere al riguardo la manual page `nice(1)`), `nice` incrementa il livello in uso al suo livello niceness 10.

Se ritenete che un'attività già in esecuzione debba essere invece avviata da `nice`, non è troppo tardi per effettuare una fix: il comando `renice` consente di modificare la priorità di un processo già in esecuzione, in positivo o in negativo (ma la riduzione della "niceness" di un processo è riservato all'utente root).

L'installazione del pacchetto `anacron` disabilita l'esecuzione di `cron` degli scripts contenuti dalle directories `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/` e `/etc/cron.monthly/`. In questo modo i suddetti scripts non saranno eseguiti due volte, sia da `anacron` che da `cron`.

`cron` rimarrà attivo e continuerà a gestire le altre attività programmate (specialmente quelle pianificate dagli utenti).

## 9.9. Quotas

Il quota system consente di circoscrivere lo spazio allocato ad un utente o ad un gruppo di utenti. Per configuararlo, è necessario disporre di un kernel che lo supporti (compilato attraverso l'opzione di configurazione `CONFIG_QUOTA`) – come fanno i kernels Debian. Il quota management system si trova nel pacchetto Debian `quota`.

Per attivare quota su un filesystem, dovete indicare le opzioni `usrquota` e `grpquota` nel file `/etc/fstab`, rispettivamente per la quota utente e per la quota del gruppo. Il riavvio del computer

aggiornerà poi le quote in assenza di attività del disco (condizione necessaria per una corretta computazione dello spazio già usato sul disco).

Il comando `edquota user` (o `edquota -g group`) vi consente di modificare i limiti mentre esamineate l'uso dello spazio sul disco corrente.

#### ANDANDO OLTRE

Come definire le "quotas" attraverso uno script

Il programma `setquota` può essere usato in uno script per cambiare automaticamente molte quotas. La manual page `setquota(8)` descrive dettagliatamente la sintassi da usare.

Il sistema quota consente di impostare quattro limiti:

- due limiti (denominati rispettivamente `soft` e `hard`) si riferiscono al numero di blocchi impiegati. Se il filesystem è stato creato con una `block size` (dimensione del blocco) di 1 kilobyte, un blocco contiene 1.024 bytes dello stesso file. Pertanto i blocchi insaturni provocano perdite di spazio sul disco. Una quota di 100 blocchi, che teoricamente consente la memorizzazione di 102.400 bytes, verrà comunque saturata con soli 100 files di 500 bytes ciascuno, per un totale di soltanto 50.000 bytes.
- due limiti (`soft` e `hard`) sono riferiti al numero di `inodes` utilizzati. Ogni file impiega almeno un inode per memorizzare le informazioni che lo riguardano (permessi, owner, data dell'ultimo accesso, ecc.). Di conseguenza attraverso questo limite viene circoscritto anche il numero di files per l'utente.

Un limite `soft` può essere superato temporaneamente; l'utente verrà semplicemente avvertito del fatto che ha superato la sua quota attraverso il comando `warnquota`, di solito invocato da `cron`. Un limite `hard` non può essere superato: il sistema rifiuterà di mettere in atto qualsiasi operazione che causi il superamento della quota `hard`.

#### DIZIONARIO

Blocks e inodes

Il filesystem suddivide l'hard drive in blocchi - piccole aree contigue. La dimensione di questi blocchi viene determinata durante la creazione del filesystem e in genere varia da 1 a 8 Kibibytes. Un blocco può essere utilizzato per archiviare i dati effettivi dei files oppure i metadati utilizzati dal filesystem. Tra questi metadati troverete in particolare gli inodes. Un inode utilizza un blocco sull'hard disk (ma il suddetto blocco non viene conteggiato fra i block quotas, bensì fra gli inode quotas) e contiene sia l'insieme delle informazioni sul file in questione (nome, owner, permessi, ecc.), sia i puntatori ai blocchi dati di fatto utilizzati. I files di notevole dimensione occupano più blocchi di quanti possano essere indicati in un singolo inode, pertanto esiste un indirect block system; l'inode si riferisce ad un elenco di blocchi che individualmente non contengono direttamente dati, ma un altro elenco di blocchi.

Attraverso il comando `edquota -t`, potrete definire un "grace period" (un "periodo di tolleranza") autorizzato e limitato, entro cui le violazioni del limite `soft` sono ancora tollerate. Trascorso il summenzionato periodo, il limite `soft` verrà considerato alla stregua di un limite `hard` e l'utente dovrà quindi ridurre lo spazio da lui usato sul disco entro il limite imposto per poter essere di nuovo in grado di scrivere sul disco.

#### ANDANDO OLTRE

Configurare una quota predefinita per i nuovi utenti

Per automatizzare la configurazione della quota per i nuovi utenti, dovete impostare un utente "modello" (attraverso `edquota` o `setquota`) ed indicare il suo nome nella variabile `QUOTAUSER` all'interno del file `/etc/adduser.conf`. La quota predefinita verrà quindi automaticamente applicata ad ogni nuovo utente creato attraverso il comando `adduser`.

## 9.10 Backup

Una delle principali responsabilità di qualsiasi amministratore è il backup e nonostante ciò rimane un argomento complesso, che implica l'uso di potenti strumenti, generalmente difficili da padroneggiare.

Esistono diversi programmi come ad esempio amanda, bacula e BackupPC. Quest'ultimi sono sistemi client/server specializzati con diverse opzioni, che possono essere piuttosto difficili da configurare. Alcuni di questi programmi supportano un'interfaccia di configurazione web user-friendly per attenuare le summenzionate difficoltà. Ma Debian contiene dozzine di altri pacchetti dedicati alle soluzioni di backup, in grado di far fronte ai casi più disparati come potrete voi stessi verificare attraverso il comando `apt-cache search backup`.

Piuttosto che trattare dettagliatamente soltanto il funzionamento di alcuni di essi, questo paragrafo descrive i pareri che influenzano i direttori dell'ipotetica Falcot Corp quando devono definire e selezionare una strategia di backup.

Alla Falcot Corp, i backups devono raggiungere due obiettivi: recuperare i files eliminati erroneamente e ripristinare rapidamente qualsiasi computer (server o desktop) il cui disco rigido è danneggiato.

[Il testo originale in inglese fa riferimento agli `use cases`, trad. lett. casi d'uso, ovvero una serie di azioni o di eventi che definiscono tipicamente le interazioni tra un ruolo (denominato attore nell'Unified Modeling Language – UML) ed un sistema indirizzate al raggiungimento di un obiettivo]

### 9.10.1. Backup con rsync

I backups su nastro sono stati considerati troppo lenti e troppo costosi, pertanto verrà eseguito il backup dei dati sui dischi rigidi di un server dedicato, dove è in uso il software RAID (andate a leggere il paragrafo 12.1.1, "Software RAID" a pag. 328) che proteggerà i dati qualora il disco si guasti. Sui computers Desktop non è stato eseguito il backup individualmente, ma gli utenti sono stati avvisati che verrà eseguito il backup del loro account personale sul file server del loro dipartimento. Il comando `rsync` (dal pacchetto omonimo) esegue quotidianamente il backup di questi diversi servers.

#### BASILARE

L'hard link, un secondo nome per il file

Un hard link (collegamento fisico), a differenza di un symbolik link (collegamento simbolico), non può essere presciso dal file "collegato". In quanto la mera creazione di un collegamento fisico di fatto corrisponde all'assegnazione di un secondo nome ad un file esistente. Questo è il motivo per cui la rimozione di un collegamento fisico rimuove solo uno dei nomi associato al file. Ovvero i dati di un file permangono nel filesystem se un altro nome è ancora assegnato al file. Occorre precisare che, a differenza di una copia, il collegamento fisico non consuma ulteriore spazio sul disco.

Il collegamento fisico viene creato con il comando `ln target link`. Il link file è pertanto un nuovo nome assegnato al target file. I collegamenti fisici possono essere creati solo all'interno dello stesso filesystem, mentre i collegamenti simbolici non sono soggetti a questa limitazione.

Lo spazio disponibile sul disco impedirebbe l'esecuzione di un backup completo giornaliero. Per tale ragione il comando `rsync` è preceduto dalla duplicazione dei contenuti dell'ultimo backup attraverso dei collegamenti fisici, che evitano di consumare troppo spazio sul disco. Dopodiché il processo `rsync` si limita a sostituire solo i files che sono stati modificati dall'ultimo backup. Questo meccanismo consente di mantenere un numero elevato di backup in una piccola quantità di spazio. Dal momento che tutti i backups sono distribuiti e resi accessibili istantaneamente (ad esempio in diverse directories dello stesso volume condiviso attraverso la rete), potrete confrontare celermemente i loro dati. Il summenzionato meccanismo di backup può essere facilmente messo in atto attraverso il programma `dirvish`. Utilizza uno spazio di archiviazione di backup (denominato sempre dal suddetto programma "bank", ossia "banca" se tradotto letteralmente in italiano) in cui inserisce le varie timestamped copies [ossia delle copie con marcatura temporale, in modo da apporre ed associare una data certa] dei sets [l'insieme, trad. lett.] dei files di backup (questi sets sono definiti "vaults" nella documentazione di "dirvish").

La configurazione principale si trova nel file `/etc/dirvish/master.conf`. Questo file definisce la posizione dell'archivio di backup, l'elenco dei "vaults" da gestire ed i valori predefiniti per la scadenza dei backups. Il resto della configurazione si trova nel file `bank/vault/dirvish/default.conf`, che contiene la configurazione specifica per il corrispondente set di files.

#### Esempio 9.3 Il file `/etc/dirvish/master.conf`

```
bank:
  /backup
exclude:
  lost+found/
  core
  */
Runall:
  root 22:00
expire-default: +15 days
expire-rule:
#  MIN HR    DOM MON        DOW  STRFTIME_FMT
  *  *      *  *          1    +3 months
  *  *      1-7 *          1    +1 year
  *  *      1-7 1,4,7,10  1
```

[ATTENZIONE! La sintassi di dirvish è differente da quella di crontab (leggete anche pag. 223). In sintesi la sintassi di dirvish:

MIN = Minute (minuto), da 0 a 59

HR = Hour (ora), da 0 a 23

DOM = Day of Month (giorno del mese), da 1 a 31

MON = Month (mese), da 1 a 12

DOW = Day of Week (giorno della settimana), da 1 a 7 (1 rappresenta la domenica e 7 il sabato)]

Il parametro `bank` indica la directory nella quale sono archiviati i backups. Il parametro `exclude` vi consentirà di definire i files (o i tipi di files) da escludere dal backup. Il `Runall` è un elenco di sets dei files di cui eseguire il backup con time-stamp per ciascun set, ovvero con assegnazione di data certa alla copia, in modo di far fronte ai casi in cui il backup non si avvia esattamente all'ora pianificata. Dovreste specificare un orario leggermente in anticipo rispetto all'orario di esecuzione effettivo (che per impostazione predefinita è impostato alle 10:04 PM sui sistemi Debian, in base a quanto previsto da `/etc/cron.d/dirvish`). Infine, i parametri `expire-default` e `expire-rule` definiscono i criteri di scadenza dei backups. Nell'esempio precedente i backups saranno per sempre generati la prima domenica di ogni trimestre; inoltre saranno distrutti dopo un anno i backups della prima domenica di ogni mese e dopo 3 mesi quelli delle altre domeniche. Gli altri backups giornalieri saranno conservati per 15 giorni. L'ordine delle regole è importante: dirvish utilizza l'ultima regola compatibile o quella definita da `expire-default` se nessuna delle altre regole menzionate in `expire-rule` risulta compatibile.

**IN PRATICA**  
Le scadenze  
programmate

Le expiration rules [trad. lett. "regole di scadenza"] non vengono impiegate da `dirvish-expire` per svolgere la propria mansione. In realtà, le expiration rules vengono applicate durante la creazione di una nuova copia di backup per definire una data di scadenza associata a quella copia. `dirvish-expire` si limita ad esaminare le copie archiviate ed elimina quelle che hanno superato la data di scadenza.

#### Esempio 9.4 Il file `/backup/root/dirvish/default.conf`

```
client: rivendell.falcot.com
tree: /
xdev: 1
```

```

index: gzip
image-default: %Y%m%d
exclude:
  /var/cache/apt/archives/*.deb
  /var/cache/man/**
  /tmp/**
  /var/tmp/**
  *.bak

```

Nell'esempio soprastante viene specificato il set di files di cui eseguire il backup: questi files si trovano sulla macchina `rivendell.falcot.com` (per un backup di dati locali, è sufficiente specificare il nome del computer locale, attraverso l'`hostname`), nel summenzionato caso in particolare quelli della root tree (`tree:/`) esclusi quelli elencati da `exclude`. Il backup rimarrà limitato al contenuto di un singolo filesystem (`xdev:1`). Non includerà i files provenienti da altri mount points. Verrà generato un elenco dei files salvati (`index:gzip`) e l'immagine verrà denominata in base alla data corrente (`image-default:%Y%m%d`).

Esistono diverse opzioni e sono tutte documentate nella manual page `dirvish.conf(5)` Una volta impostati questi files di configurazione, ogni set di files dovrà essere inizializzato attraverso il comando `dirvish --vault vault --init`. Giunti a questo punto l'invocazione giornaliera di `dirvish-runall` creerà automaticamente una nuova copia di backup subito dopo aver eliminato le copie di backups che devono essere cancellate.

[Genericamente: il termine *hooking* (trad. lett. "agganciare") viene utilizzato in riferimento ad una serie di tecniche utilizzate per alterare o accrescere le funzionalità di un sistema operativo, di un'applicazione o di altri componenti software intercettando chiamate, messaggi, eventi scambiati tra le componenti del software attraverso un codice denominato "hook"; il protocollo LDAP (Lightweight Directory Access Protocol) è un protocollo applicativo standard (open e vendor-neutral) per l'accesso e la manutenzione del directory service su una rete IP (Internet Protocol); un directory service o name service è un servizio che "mappa" (associa) i nomi delle risorse di rete ai loro indirizzi di rete.]

#### IN PRATICA

Remote backup  
attraverso SSH

Qualora `dirvish` dovesse salvare i dati in una macchina remota, utilizzerà `ssh` per connettersi ed avviare `rsync` per svolgere le sue mansioni come fa in un normalissimo server. Quanto sopra espresso richiede pertanto che l'utente root sia in grado di connettersi automaticamente alla macchina in questione. In casi come questo entra in gioco l'impiego di una chiave di autenticazione SSH (andate a leggere al riguardo il paragrafo 9.2.1.1, "Autenticazione basata su chiave" a pagina 208).

### 9.10.2. Ripristino delle macchine senza Backups

I computers Desktop, privi di un backup, potranno essere reinstallati da DVD-Rom personalizzati realizzati attraverso Simple-CDD (andate a leggere il paragrafo 12.3.3, "Simple-CDD: la soluzione all-in-one" a pagina 370). Dal momento che si tratta di un installazione da zero, perderete qualsiasi configurazione abbiate effettuato dopo l'installazione iniziale. Nonostante ciò rimane comunque un metodo accettabile dato che: tutti i sistemi vengono "hooked" ad un centrale LDAP directory per gli accounts e la maggior parte delle applicazioni desktop vengono preconfigurate mediante dconf (andate a leggere il paragrafo 13.3.1, "GNOME" a pagina 385 per maggiori informazioni sull'argomento).

Gli amministratori della Falcot Corp sono a conoscenza dei limiti della loro politica di backup. Di conseguenza, non essendo in grado di proteggere il server di backup con la stessa efficienza di una cassaforte ignifuga (se al suo interno fosse custodito un nastro) hanno preferito installare il suddetto server di backup in una stanza separata, così che un evento accidentale, come un incendio nella stanza dei servers, non possa distruggere anche i backups. Inoltre, eseguono una volta alla settimana un backup incrementale su DVD-Rom coinvolgendo solo i files modificati dall'ultimo backup.

#### CULTURA

TAR, standard di  
backup su nastro

Storicamente, la creazione di un archivio TAR su nastro era il modo più semplice per eseguire un backup Unix. Difatti lo stesso comando `tar` prende il suo nome dalla locuzione "Tape Archive".

---

## ANDARE OLTRE

### Backup dei servizi SQL e LDAP

Non è possibile effettuare il backup di alcuni servizi (come ad esempio i databases SQL o LDAP) attraverso la mera copia dei loro files (a meno che non li interrompiate adeguatamente durante il backup, cosa alquanto problematica in quanto sono designati per essere sempre accessibili). Dovrete quindi eseguire una procedura di "export" per creare un "data dump" (trad. lett. "scaricamento dati") di cui potrete eseguire il backup in sicurezza. Spesso i "data dumps" sono piuttosto "ingombranti", ma si prestano bene ad essere compressi. Per ridurre lo spazio di archiviazione a loro necessario, dovrete salvare solo un text file alla settimana ed un diff al giorno; potrete creare il diff attraverso il comando `diff file_di_ieri file_di_oggi`. Il programma `xdelta` produrrà le differenze incrementalali dai binary dumps.

## 9.11. Hot Plugging: hotplug

### 9.11.1. Introduzione

Il sottosistema del kernel hotplug gestisce dinamicamente la connessione e la rimozione dei dispositivi, per mezzo del caricamento degli opportuni drivers e della creazione dei corrispondenti device files (con l'ausilio di udevd). Il moderno hardware, nonché la virtualizzazione consentono a quasi tutti i dispositivi di essere "hotplugged" [connessi a "caldo"]: dalle classiche periferiche USB/PCMCIA/IEEE 1394 ai dischi rigidi SATA, senza dimenticare la CPU e la memoria.

Il kernel possiede un database che associa a ciascun device ID il driver necessario. Il suddetto database viene utilizzato durante il boot per caricare tutti i drivers sia delle periferiche rilevate sui differenti buses, sia dei dispositivi hotplug quando effettuate la loro connessione alla macchina. Dopodiché, nel momento in cui il dispositivo diventa pronto per essere utilizzato, viene inviato a udevd un messaggio, che lo autorizza a creare una voce correlata in /dev/.

### 9.11.2. Il problema dell'assegnazione dei nomi

Prima dell'introduzione delle connessioni hotplug, veniva assegnato un nome fisso ai dispositivi. Ci si basava semplicemente sul posizionamento delle periferiche sui loro rispettivi bus. Ma tale sistema non era idoneo con le periferiche che venivano connesse e sconnesse dai buses. Casi tipici di questi dispositivi sono le fotocamere digitali e le chiavette USB, rilevati dal computer come disk drives. Di conseguenza, ad esempio, al primo dispositivo connesso [fra i due sopraccitati] potenzialmente potrebbe essere assegnato /dev/sdb, al secondo /dev/sdc (mentre /dev/sda viene assegnato per impostazione predefinita al disco rigido interno del computer). Pertanto non è possibile un assegnazione fissa del nome (al dispositivo), in quanto il nome in sé dipende dall'ordine in cui sono stati collegati i dispositivi.

Inoltre, sempre più drivers utilizzano i dynamic values per i numeri major/minor dei dispositivi, il che rende impossibile utilizzare una voce statica per il dispositivo, poiché queste caratteristiche essenziali possono variare dopo il riavvio del computer. [Genericamente: il numero "major", comune per tutti i dispositivi controllati dallo stesso driver, identifica per il kernel il tipo di dispositivo; il numero "minor" invece identifica per il driver le caratteristiche peculiari del dispositivo in modo da renderle accessibili.] Per risolvere questi problemi è stato quindi creato udev.

### 9.11.3. Come funziona udev

Quando udev viene informato dal kernel della presenza di un nuovo dispositivo, recupera le diverse informazioni sul dispositivo in questione consultando le voci corrispondenti in /sys/, in particolare quelle che consentono di identificarlo univocamente (indirizzo MAC per una scheda di rete, numero seriale per alcuni dispositivi USB, ecc.).

[A seguire tavola estremamente generica e non presente nel testo originale in inglese]

Simboli operatori di confronto	Significato	Esempio pratico
<code>==</code>	uguale a	<code>x==y</code>
<code>!=</code>	diverso da	<code>x!=y</code>
<code>&lt;</code>	minore	<code>x&lt;y</code>
<code>&gt;</code>	maggiore	<code>x&gt;y</code>
<code>&lt;=</code>	minore o uguale	<code>x&lt;=y</code>
<code>&gt;=</code>	maggiore o uguale	<code>x&gt;=y</code>

Provveduto di tutte queste informazioni, udev consulta l'insieme di regole contenute in `/etc/udev/rules` e `/lib/udev/rules.d/`. Durante questo processo udev decide quale nome assegnare al dispositivo, che collegamenti simbolici creare (per impartirgli nomi alternativi), nonché quali comandi eseguire. Tutti i files vengono consultati e le regole vengono valutate sequenzialmente (tranne quando un file utilizza delle direttive "GOTO"). In questo modo potrebbero esserci diverse regole che corrispondono ad un dato evento.

La sintassi dei rules files è abbastanza semplice: ogni riga contiene dei selection criteria e variable assignments. Il selection criteria consente di selezionare gli eventi per cui occorre "reagire" e il variable assignment definisce "l'azione" da eseguire in conseguenza agli eventi. Questi sono semplicemente separati da virgole ed è l'operatore che implicitamente distingue i criteri di selezione (ad esempio, attraverso gli operatori di confronto `==` o `!=`) dalle direttive di assegnamento (ad esempio, attraverso gli operatori `=`, `+=` o `:=`). Gli operatori di confronto sono usati sulle seguenti variabili:

- KERNEL: il nome che il kernel assegna al dispositivo;
- ACTION: l'azione corrispondente all'evento ("add" quando viene connesso un dispositivo, "remove" quando viene disconnesso un dispositivo);
- DEVPATH: il percorso della voce corrispondente al dispositivo in `/sys/`;
- SUBSYSTEM: il sottosistema del kernel che genera la richiesta (sono numerosi, ma qui citiamo solo pochi esempi "usb", "ide", "net", "firmware", ecc.);
- ATTR{attribute}: file contents dell'attribute file nella directory `/sys/$devpath/` del dispositivo. Qui è dove troverete gli indirizzi MAC ed altri identificatori specifici per ciascun bus; [Genericamente: i file attributes sono un tipo di metadati che descrivono i files/directories e attraverso cui è possibile modificare il funzionamento di files/directories in un filesystem; il file contents è una funzione per l'accesso ai dati, riferita ai contenuti del file e viene usata per le ricerche avanzate.]
- KERNELS, SUBSYSTEMS e ATTRS{attributes} sono delle variabili che si armonizzeranno (o almeno tenteranno) con le diverse opzioni dei parent devices del dispositivo corrente;
- PROGRAM: delega il test ad un programma indicato (true se restituisce 0, altrimenti false). I contenuti dello standard output del programma vengono conservati per essere riutilizzati nel RESULT test;
- RESULT: esegue tests sullo standard output memorizzato durante l'ultima chiamata a PROGRAM.

Gli operandi a destra possono utilizzare pattern expressions per abbinare più valori contemporaneamente. Per esempio, `*` corrisponde a qualsiasi stringa (anche se vuota), `?` corrisponde a qualsiasi carattere e `[]` corrisponde all'insieme di caratteri elencati tra le parentesi quadre (altrimenti se il primo carattere tra le parentesi quadre è un punto esclamativo i caratteri successivi elencati sempre all'interno delle parentesi quadre vengono esclusi – gli intervalli di caratteri contigui seguono l'ordine alfabetico dalla a alla z).

Riguardo agli operatori di assegnazione, `=` assegna un valore (e sostituisce il valore corrente); qualora si tratti di un elenco, questi viene svuotato e sostituito con solo il valore assegnato. `:=` fa lo stesso ma impedisce successive modifiche alla stessa variabile. `+=` aggiunge un elemento in un elenco. Le seguenti variabili possono essere modificate:

- NAME: il filename del dispositivo da creare in `/dev/`. Conta solo il primo assignment, gli altri vengono ignorati;
- SYMLINK: l'elenco dei collegamenti simbolici che punteranno allo stesso dispositivo;

[Tavola estremamente generica e non presente nel testo originale in inglese -  
segue anche sulla pagina successiva]

Esempio operatori di assegnamento	Significato
<code>x=y</code>	Memorizza il valore del secondo operando nell'oggetto specificato dal primo operando (assegnamento semplice).
<code>x**=y</code>	Moltiplica il valore del primo operando per il valore del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x/=y</code>	Divide il valore del primo operando per il valore del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x%*=y</code>	Calcola il modulo (valore assoluto) del primo operando eseguendo la divisione del valore del secondo operando per il valore del primo operando. Il risultato deve essere un numero intero e non ammette resti. In caso contrario viene considerato dall'interprete come risultato il valore assoluto del primo operando. Il risultato viene memorizzato nell'oggetto del primo operando.

<code>xdiv=y</code>	Divide il valore del primo operando per il valore del secondo operando e memorizza il risultato (ottiene un numero intero) nell'oggetto del primo operando.
<code>x+=y</code>	Somma il valore del secondo operando al valore del primo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x-=y</code>	Sottrae il valore del secondo operando dal valore del primo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x&lt;&lt;=y</code>	Esegue uno spostamento a sinistra del valore del primo operando di un numero di bit specificati dal valore del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x&gt;&gt;=y</code>	Esegue uno spostamento a destra del valore del primo operando di un numero di bit specificati dal valore del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x&amp;=y</code>	Ottiene l'AND logico bit per bit del primo e del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x^=y</code>	Ottiene l'OR logico esclusivo (XOR) bit per bit del primo e del secondo operando e memorizza il risultato nell'oggetto del primo operando.
<code>x  =y</code>	Ottiene l'OR logico inclusivo bit per bit del primo e del secondo operando e memorizza il risultato nell'oggetto del primo operando.

- OWNER, GROUP e MODE definiscono l'user ed il group titolari dei diritti del dispositivo così come i correlati permessi;
- RUN: l'elenco dei programmi da eseguire in risposta a questo evento.

I valori assegnati a queste variabili potrebbero impiegare delle sostituzioni:

- \$kernel o %k: equivalente al KERNEL;
- \$number o %n: il numero di posizione del dispositivo [nell'ordine dei dispositivi connessi], ad esempio per sda3 il numero "3";
- \$devpath o %p: equivalente a DEVPATH;
- \$attr{attribute} o %s{attribute}: equivalente a ATTRS{attribute};
- \$major o %M: il kernel major number del dispositivo;
- \$minor o %m: il kernel minor number del dispositivo;
- result o %c: la string output dell'ultimo programma invocato da PROGRAM;
- infine i caratteri % e \$\$ rispettivamente per i caratteri percentuale e dollaro.

Queste liste non sono esaustive (includono i parametri più importanti) ma potrete trovare un elenco approfondito nella manual page udev(7).

#### 9.11.4. Un caso concreto

Prendiamo in considerazione il caso di una semplice chiavetta USB e che desideriate provare ad assegnargli un nome fisso. Dovrete innanzitutto trovare gli elementi che ne consentiranno l'identificazione in modo univoco. Per fare ciò, collegate il dispositivo ed eseguite udevadm info -a -n /dev/sdc (ovviamente sostituendo al bisogno /dev/sdc con il nome corrente assegnato alla chiave usb).

```
# udevadm info -a -n /dev/sdc
[...]
looking at device '/devices/pci0000:00/0000:00:10.0/usb2/2-1/2-1:1.0/host4/target4
-> :0:0/4:0:0:0/block/sdc':
  KERNEL=="sdc"
  SUBSYSTEM=="block"
  DRIVER==""
  ATTR{hidden}=="0"
  ATTR{events}=="media_change"
  ATTR{ro}=="0"
  ATTR{discard_alignment}=="0"
  ATTR{removable}=="1"
  ATTR{events_async}==""
  ATTR{alignment_offset}=="0"
  ATTR{capability}=="51"
  ATTR{events_poll_msecs}=="-1"
  ATTR{stat}==" 130 0 6328 435 0 0 0
-> 0 0 252 252 0 0 0 0"
  ATTR{size}=="15100224"
  ATTR{range}=="16"
```

```

ATTR{ext_range}=="256"
ATTR{inflight}=="0"          0
[...]
looking at parent device '/devices/pci0000:00/0000:00:10.0/usb2/2-1/2-1:1.0/host4/
-> target4:0:0:4:0:0:0':
[...]
ATTRS{max_sectors}=="240"
[...]
looking at parent device '/devices/pci0000:00/0000:00:10.0/usb2/2-1':
KERNELS=="2-1"
SUBSYSTEMS=="usb"
DRIVERS=="usb"
ATTRS{bDeviceProtocol}=="00"
ATTRS{bNumInterfaces}==" 1"
ATTRS{busnum}=="2"
ATTRS{quirks}=="0x0"
ATTRS{authorized}=="1"
ATTRS{ltm_capable}=="no"
ATTRS{speed}=="480"
ATTRS{product}=="TF10"
ATTRS{manufacturer}=="TDK LoR"
[...]
ATTRS{serial}=="07032998B60AB777"
[...]

```

Per creare una nuova regola, è possibile utilizzare i tests sulle variabili del dispositivo o su uno dei parent devices. L'esempio soprastante vi permette di creare due regole come le seguenti:

```

KERNEL=="sd?", SUBSYSTEM=="block", ATTRS{serial}=="07032998B60AB777", SYMLINK+="
-> usb_key/disk"
KERNEL=="sd?[0-9]", SUBSYSTEM=="block", ATTRS{serial}=="07032998B60AB777", SYMLINK+="
-> usb_key/part%n"

```

Una volta che queste regole sono state configurate in un file, chiamato ad esempio /etc/udev/rules.d/010\_local.rules, dovete solo rimuovere e reinserire la chiave USB. Potrete quindi verificare che /dev/usb\_key/disk rappresenta il disco associato alla chiave USB, mentre /dev/usb\_key/part1 la sua prima partizione.

#### ANDANDO OLTRE Debugging della configurazione di udev

Come molti demoni, udevd registra i logs in /var/log/daemon.log. Ma non essendo molto “verboso” [trad. non lett. “dettagliato”] per impostazione predefinita, raramente si riesce a capire cosa stia accadendo. Per risolvere questo problema eseguite il comando udevadm control --log-priority=info in quanto incrementa il livello di verbosità corrente. udevadm control --log-priority=err ripristina il livello di verbosità predefinito.

## **9.12 Power Management: Advanced Configuration and Power Interface (ACPI)**

La questione del power management [gestione dell'energia] è spesso complessa. Difatti, una sospensione consona richiede che tutti drivers di tutti i devices del computer siano in grado di porre i devices stessi in standby [modalità d'attesa] e di riconfigurare i correlati devices quando il computer riprende l'attività. Sfortunatamente, ci sono ancora alcuni devices che non sono in grado di entrare correttamente in sleep mode sotto Linux, perché i loro produttori non hanno supportato le specifiche necessarie.

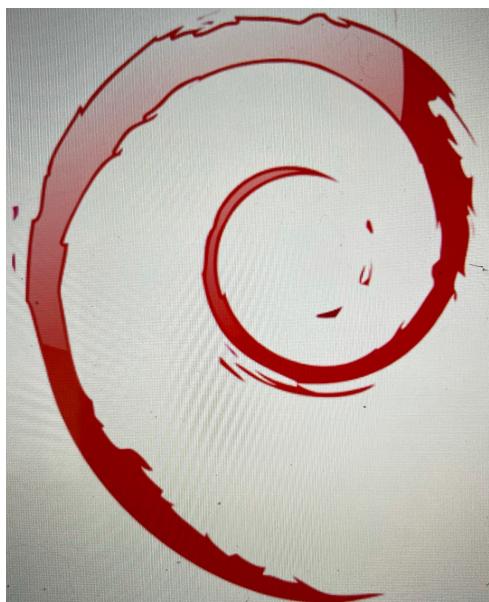
Linux supporta ACPI (Advanced Configuration and Power Interface), lo standard più recente in power management. Il pacchetto acpid offre un demone che verifica gli eventi relativi alla gestione dell'energia (ad esempio il passaggio in un laptop dalla batteria all'alimentazione AC) e che può eseguire vari comandi in risposta.

### **FARE ATTENZIONE**

Scheda Grafica e  
standby

Il driver della scheda grafica è spesso l'origine dei malfunzionamenti dello standby. Se siete afflitti da un'anomalia, dovreste verificare se l'ultima versione del server grafico X.org è in grado di risolvere il problema.

Dopo questa panoramica dei servizi base comuni nei sistemi Unix, ci concentreremo sull'ambiente in cui operano le macchine amministrate ovvero la rete. Molti servizi sono infatti necessari perché la rete funzioni correttamente. Quest'ultimi, in particolare, saranno trattati nel capitolo successivo.



---

## Parole chiave

Network  
Gateway  
TCP/IP  
IPv6  
DNS  
Bind  
DHCP  
QoS

