1 Linux Networking

1.1 Indirizzi MAC

Assegnati dal produttore della scheda. Numeri di 48 bit. Numero di lotto - numero di scheda. Spesso è possibile cambiarli via software.

1.2 Indirizzi IP

Dato che gli indirizzi IP in Internet devono essere univoci, è meglio usare come indirizzi per i collegamenti in rete locale gli indirizzi privati Internet:

Un indirizzo della rete di classe A **privata**:

10.0.0.0 (netmask 255.0.0.0)

Un indirizzo in una delle 16 reti private di classe B:

da 172.16.0.0 a 172.31.0.0 (netmask 255.255.0.0)

Un indirizzo in una delle 256 reti private di classe C:

da 192.168.0.0 a 192.168.255.0 (netmask 255.255.255.0)

1.3 "Interfacce" di rete

Ogni scheda di rete (interfaccia) deve avere almeno un indirizzo IP. Il nome dell'interfaccia evoca la tecnologia ed il numero d'ordine della scheda.

lo (loopback) 127.0.0.1; eth0, eth1 .. , ppp0, ppp1, ..

Se si vuole configurare una scheda in modo che abbia più di un indirizzo IP (aliasing IP), si definiscono interfacce "derivate". Es. eth0:1, eth0:2, ..

Le interfacce ppp0, ppp1, .. sono associate ai modem e sono attive solo quando si è collegati alla Rete via modem acustico.

1.4 ping

Programma diagnostico per i livelli MAC e IP delle rete. Se il ping funziona, funzionano i livelli MAC e IP.

ping broadcast

```
$ ping -b 192.178.13.255
```

Oggi moltissimi host di rete sono configurati per non rispondere ai ping broadcast, perché nel passato questo strumento è stato abusato per attaccare i sistemi in rete e causare Denial o Service (Dos).

1.5 ifconfig (interface configure)

ifconfig è il comando che permette di configurare le interfacce di rete. Se usato senza alcun parametro mostra l'attuale configurazione di tutte le interfacce di rete.

Esempio:

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          inet addr:192.168.13.10  Bcast:192.168.13.63  Mask:255.255.255.192
          inet6 addr: fe80::214:38ff:fe0a:43ca/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:176 (176.0 b)
          Interrupt:11

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:3600 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3600 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:4692270 (4.4 MiB)  TX bytes:4692270 (4.4 MiB)
```

In questo esempio si vede la configurazione corrente di due interfacce: "eth0" (la "zeresima" scheda di rete Ethernet), e "lo" ("loopback", un'interfaccia "virtuale", sempre presente, che permette di usare il software TCP/IP per collegare programmi che risiedono sullo stesso computer).
ifconfig visualizza: l'indirizzo MAC della scheda di rete, l'indirizzo IP che le è stato assegnato (inet addr) l'indirizzo che la scheda userà per il broadcast nella sottorete (Bcast), la maschera di sottorete (), l'indirizzo IP v6 (), e le statistiche d'uso della interfaccia di rete: pacchetti (frame) trasmessi e ricevuti, pacchetti persi e sovrascritti (overruns, dovuto a collisioni non rilevate), collisioni avvenute, occupazione massima del buffer di trasmissione.
La sintassi generale di ifconfig è la seguente:

 # ifconfig <interfaccia> <indirizzo> <opzioni>

Esempio:

```
# ifconfig eth0 192.168.13.10/26
```

Configura per l'interfaccia eth0 l'indirizzo indicato. La netmask è quella che corrisponde alla classe IP dell'indirizzo, l'indirizzo di broadcast è quello con tutti 1 nella parte di host, considerando la maschera di default.

Si può anche dare una maschera di sottorete specifica, indicando il prefisso CIDR o la maschera stessa, con la sintassi che si evince dai seguenti esempi:

```
# ifconfig eth0 192.168.13.10 netmask 255.255.255.192
```

equivale a

```
# ifconfig eth0 192.168.13.10/26
```

ifconfig non è "persistente" fra un boot e l'altro, le modifiche fatte con ifconfig si perdono. ifconfig viene chiamato al boot da uno degli script .rc ed imposta gli indirizzi che sono scritti nello script stesso.
Per rendere definitive le configurazioni provate con ifconfig bisogna modificare gli script .rc, direttamente o, molto meglio, per mezzo di un programma grafico di configurazione della rete, sempre presente nelle varie distribuzioni Linux.

E' importante far notare che ifconfig non permette di stabilire il default gateway per l'interfaccia di rete che si configura. Questo si potrà (e si dovrà) fare successivamente, inserendo nella tabella di routing del computer una route di default, con il comando route (vedi oltre).

Opzioni interessanti di ifconfig

```
# ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```

imposta, se il driver della interfaccia di rete lo consente, l'indirizzo MAC indicato (nell'esempio AA:BB:CC:DD:EE:FF), diverso da quello assegnato alla scheda dal suo produttore.

alias IP

Il comando ifconfig permette anche di configurare gli alias IP, cioè indirizzi IP alternativi per la stessa interfaccia di rete. Per far ciò basta impostare l'interfaccia virtuale ethX:Y, che significa l'Y alias dell'interfaccia X (es. eth0:0, eth0:1 ..)
Esempio:

```
# ifconfig eth0 192.168.13.25
# ifconfig eth0:0 10.11.12.13
# ifconfig eth0:1 172.16.13.8
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          inet addr:192.168.13.25  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::214:38ff:fe0a:43ca/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:173 errors:1 dropped:1 overruns:0 frame:0
          TX packets:119 errors:0 dropped:0 overruns:0 carrier:0
          collisions:9 txqueuelen:1000
          RX bytes:39433 (38.5 KiB)  TX bytes:11190 (10.9 KiB)
          Interrupt:11

eth0:0    Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          inet addr:10.11.12.13  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:11

eth0:1    Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          inet addr:172.16.13.8  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
          Interrupt:11
```
Come si vede, i tre comandi ifconfig hanno configurato tre interfacce IP virtuali sulla stessa scheda di rete, sull'interfaccia "principale" (eth0) è impostato 192.168.13.25, sull'interfaccia virtuale eth0:0 è 10.11.12.13 mentre sull'interfaccia virtuale eth0:1 è impostato 172.16.13.8.
Usare uno qualsiasi dei tre indirizzi sulla stessa scheda è del tutto equivalente.

```
# ifconfig eth0:1 down
```
"Spegne" l'interfaccia virtuale eth0:1, disabilitando l'alias IP 172.16.13.8; gli altri alias e l'IP "principale" dell'interfaccia continuano a funzionare.

```
# ifconfig eth0 down
```
"Spegne" l'interfaccia eth0, trattandosi dell'IP "principale" anche gli alias vanno "down".

ifup, ifdown
Questi comandi attivano (ifup) o disattivano (ifdown) le interfacce di rete che vengono specificate come parametro.
Esempio:

```
# ifdown eth0
```
Disattiva l'interfaccia di rete, che ora non funziona più, infatti se faccio ifconfig, ottengo:
```
# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:3690 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3690 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4786983 (4.5 MiB)  TX bytes:4786983 (4.5 MiB)
```

Come si vede c'è solo l'interfaccia di loopback, eth0 ora non "esiste".

ifconfig si può usare con l'opzione down o up, per "spegnere" o accendere anche le interfacce virtuali.
Esempi:
```
# ifconfig eth0:1 down
```

1.6 Programma ipcalc

E' un calcolatore di indirizzi IP, vuole sempre un indirizzo IP, seguito eventualmente da una maschera di sottorete o da un prefisso CIDR. Non è sempre installato da tutte le distribuzioni.
Esempi:
```
# ipcalc -b 37.243.122.1/21
BROADCAST=37.243.127.255
```
l'opzione broadcast dà l'indirizzo IP di broadcast per la rete (indirizzo e netmask) passata

```
# ipcalc -m 37.243.122.1/21
NETMASK=255.255.248.0
```
l'opzione m (mask) dà la netmask per la rete passata

```
# ipcalc -p 0.0.0.0 255.255.248.0
PREFIX=21
```
La precedente serve per sapere il prefisso di una netmask scritta in forma "a quattro punti"

```
# ipcalc -pn 37.243.122.1 255.255.248.0
PREFIX=21
NETWORK=37.243.120.0
```
L'opzione p (prefix) dà il prefisso che corrisponde alla maschera di sottorete passata a ipcalc
L'opzione n (network) dà la parte di rete dell'indirizzo + maschera passati
Come si vede dall'ultimo esempio, le opzioni possono essere passate insieme.

1.7 Porte TCP/UDP e well known port number

Il file **/etc/services** contiene una tabella che mostra la corrispondenza fra servizi-protocolli Internet e numeri di porta TCP-UDP. I demoni che realizzano i protocolli Internet leggono da questo file per stabilire i loro port di default.
Le porte con numero inferiore a 1024 ("well known port number") possono essere usate solo da root, in questo modo i client possono essere sicuri che i servizi che utilizzano non siano lanciati da un utente qualsiasi del sistema.
Il formato del file è:

<nome del servizio> <porta in decimale>/<tcp | udp> <alias> #<commenti>

1.8 arp

<alias> è una lista di nomi alternativi per il protocollo.
Address Resolution Protocol serve agli host per trovare l'indirizzo MAC di un host del quale conoscono l'indirizzo IP.
Per vedere il contenuto attuale della cache ARP digitare:

```
# arp -a
```

Per evitare ARP nelle reti con schede che non cambiano mai: scrivere nel file /etc/ethers

1.9 Client DHCP

Tramite DCHP è possibile conoscere, oltre l'indirizzo IP, anche molti altri parametri per il funzionamento in rete di un host. Una Linux box può contattare, in broadcast sulla rete locale, un server DHCP per ottenere questi parametri, che userà "temporaneamente", per il tempo che le verrà indicato.
Il più "standard" fra i software che permettono di ottenere un indirizzo IP tramite il protocollo DHCP è "dhclient", prodotto dall'"Internet Software Consortium".
Se il sistema Linux è già "preconfigurato", per fare una richiesta DHCP potrebbe bastare lanciare il programma dhclient ed attendere (un po'!) per la risposta del server.

```
# dhclient
```

dhclient fa la richiesta DHCP, poi si mette a funzionare in background, come un demone, controllando quando ci sono dei cambiamenti nell'indirizzo IP o negli altri parametri della connessione.

L'opzione -r (release) di dhclient "restituisce" al server l'indirizzo e disattiva l'interfaccia.
Esempio:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          inet addr:192.168.1.3  Bcast:192.168.1.31  Mask:255.255.255.224
          inet6 addr: fe80::214:38ff:fe0a:43ca/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             ..
# dhclient -r
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:14:38:0A:43:CA
          BROADCAST MULTICAST  MTU:1500  Metric:1
             ..
```

Dove si vede che l'interfaccia non ha più un indirizzo IP.

Un esempio specifico:
Quando il programma di amministrazione delle schede di rete lancia il client DHCP su un sistema Fedora Core 4 fa qualcosa del genere:

```
# /sbin/dhclient -1 -q -lf /var/lib/dhcp/dhclient-eth0.leases -pf /var/run/dhclient-eth0.pid eth0
```

Dove:

-1 fa un solo tentativo, se non trova un server DHCP, rinuncia a configurare l'interfaccia
-q non mostra informazioni sulla console al momento dell'avvio
-lf (**l**ease **f**ile) è l'opzione che indica il "lease file", in questo caso il nome del file è "personalizzato" per 'interfaccia eth0, il file "standard" sarebbe "/var/lib/dhcp/dhclient.leases"
-pf (**P**ID **f**ile) è il file che contiene il numero di identificazione (PID) del processo che fa eseguire il programma dhclient

l'ultimo parametro dell'esempio mostrato è l'interfaccia di rete sulla quale viene fatta la richiesta DHCP (eth0 nell'esempio). Se manca l'indicazione dell'interfaccia, vengono "provate" tutte le interfacce di rete che sono in grado di fare il broadcast.
Se fosse presente un'opzione -cf (configuration file) si specificherebbe un file di configurazione alternativo, diverso da quello "standard" (che è "dhclient.conf", di solito in /etc od in uno dei suoi "figli").
Il file dhclient.leases viene scritto dal programma dhclient e tiene traccia degli indirizzi che sono stati assegnati al computer. Esso viene usato, per esempio, quando il computer viene fatto ripartire. Può contenere qualcosa del genere:

```
lease {
  interface "eth0";
  fixed-address 172.16.0.16;
  option subnet-mask 255.255.0.0;
  option dhcp-lease-time 691200;
  option routers 172.16.0.254;
  option dhcp-message-type 5;
  option dhcp-server-identifier 172.16.0.253;
```

```
  option domain-name-servers 172.16.0.253;
  option dhcp-renewal-time 345600;
  option dhcp-rebinding-time 604800;
  renew 1 2006/3/13 04:51:33;
  rebind 4 2006/3/16 13:22:25;
  expire 5 2006/3/17 13:22:25;
}
```

Nel file ci saranno una o più sezioni del genere, una per ogni "lease" ("prestito") che il nostro computer avrà ottenuto su ogni sua interfaccia coinvolta nel DHCP.
"fixed-address" è l'indirizzo IP che viene dato all'interfaccia. Sarà "fisso" al massimo per il tempo "dhcp-lease-time", indicato oltre. Oltre tale tempo sarà "ritirato" dal server; prima di quel momento potrà essere "rinnovato", a partire dal "dhcp-renewal-time".
Nell'esempio precedente si vedono altri parametri dell'interfaccia, che si commentano da soli in base al loro nome.
Il "dhcp-lease-time" dell'esempio è di 691200 s, corrispondenti ad 8 giorni, il "dhcp-renewal-time" è 345600 s, corrispondenti a 4 giorni, il "dhcp-rebinding-time"[1] è di 604800 s, corrispondenti a 7 giorni. Dalle ultime righe della sezione si deduce che il lease è stato iniziato il giorno 9 Marzo 2006.

Configurazione del client DHCP
Il programma può essere configurato in modo molto esteso, attraverso il file dhclient.conf; alcune configurazioni interessanti sono:

- request (richieste): le richieste al server di particolari informazioni sulla rete
  - il server può fare la richiesta di uno specifico indirizzo o di un certo tempo di lease, che il server potrà accordare o negare.
- require (requisiti): le informazioni "obbligatorie" che il client deve acquisire. Se un server non darà tutte queste informazioni, nell'ordine specificato, il client non acquisirà l'indirizzo
- lease ("affitti"): richieste di indirizzi
  - lease fissi (fixed-address): si può configurare un lease con un indirizzo specifico
    - quando il client non trova un server prova tutti i server DHCP da cui ha avuto in passato un indirizzo che risulti ancora valido. Se non ne trova nessuno si può configurare per assumere un indirizzo fisso, in un lease che non finisce mai.
- renew time: il momento in cui il client comincia a ricontattare il server per ottenere il rinnovo dell'autorizzazione all'uso dell'indirizzo
- rebind time: stabilisce il momento in cui il client deve cominciare a cercare un qualsiasi server DHCP, perché il lease sta per scadere e bisogna cercare un indirizzo in broadcast
- expire time: è il momento in cui l'indirizzo non vale più; il client deve cessare l'utilizzazione dell'indirizzo.

1.10 Risoluzione di nomi in indirizzi

"Risolvere" un nome significa trovare un indirizzo IP che corrisponda ad un simbolo (una stringa). Per esempio al nome "goku" potrebbe corrispondere l'indirizzo IP 192.168.1.22 .

File /etc/host.conf

Il contenuto di questo file determina il comportamento del sistema nella risoluzione dei nomi.
Tipicamente c'è scritto:

```
order hosts, bind
multi on
```

che significa che per risolvere i nomi il sistema cerca prima nel file /etc/hosts, poi chiede in rete, ai server DNS specificati in /etc/resolv.conf (bind è il nome del demone che fa da nameserver).
multi on significa che vengono considerati tutti gli indirizzi ricavati precedentemente e non solo il primo trovato. Potrebbe mancare in host.conf.

File /etc/hosts

Questo file stabilisce delle corrispondenze fra indirizzi IP e nomi simbolici (stringhe). In ogni riga del file sta un indirizzo IP ed uno o più nomi corrispondenti. Nella risoluzione dei nomi /etc/hosts viene di solito consultato PER PRIMO (vedi "order hosts, bind"), per cui se un nome è in questo file non verrà richiesto al DNS.

In Windows esiste un file corrispondente:

C:\Windows\hosts.sam, nei sistemi Win 9X-Me, mentre c'è C:\WINNT\system32\drivers\etc\hosts in NT-XP.

1 È il tempo oltre il quale il computer comincia a cercare un nuovo indirizzo da qualsiasi server DHCP, anche diverso da quello che gli ha dato l'indirizzo precedentemente.

File /etc/resolv.conf
Il file /etc/resolv.conf specifica il comportamento del sistema nella risoluzione dei nomi degli host con il protocollo DNS. Tipicamente è composto di tre righe; una contiene il nome del dominio (DNS) del computer, un'altra una lista di nomi di dominio.
nameserver <Indirizzo IP>

è un singolo indirizzo IP, che indica il server DNS cui chiedere gli indirizzi corrispondenti ai nomi che non si riescono a risolvere "all'interno del computer" (con /etc/hosts).
Si possono mettere più linee come nameserver, esse verranno scandite nell'ordine con cui sono scritte in questo file.

search <Lista di nomi di dominio>

è una lista di nomi di dominio, separati da spazi.
Indica in quali dominii cercare per trovare un host il cui nome non sia completamente specificato.

domain <Nome di dominio>

è un singolo nome di dominio, con le convenzioni del DNS.
Dice come completare il nome degli host che non siano completamente specificati.

1.10.1 Programmi per il lookup DNS

Il programma nslookup fa un'interrogazione al DNS, la sua sintassi è semplice:

```
# nslookup <nome di dominio>
```

dà l'indirizzo IP del <nome di dominio>

Il programma host fa un'interrogazione invertita al DNS (reverse lookup):

```
# host <indirizzo IP>
```

dà il nome di dominio DNS cui appartiene l'<indirizzo IP>.
Il programma host permette anche di fare un lookup "normale":

```
# host <nome di dominio>
```

1.10.2 Programma netstat

Mostra le statistiche di utilizzazione della rete, le tabelle di routing, le connessioni internet attive, le connessioni NAT.

Esempi:

```
# netstat
```
mostra i socket aperti, TCP e UDP. I socket TCP vengono indicati come "STREAM", quelli UDP come "DGRAM" (datagram). Non visualizza i socket dei server in ascolto; se si vogliono visualizzare anch'essi, bisogna dare l'opzione -a (all).

```
# netstat -i
```
-i = interface: mostra le statistiche di traffico relative alle interfacce di rete.

```
# netstat -M
```
-M = Masquerade: mostra le sessioni TCP che sono state "mascherate" da indirizzo privato ad indirizzo pubblico.

1.11 Routing: comando route

La tabella di routing in Linux è **/proc/net/route**.
Il contenuto della tabella si può vedere visualizzando il "file", oppure usando i comandi route e netstat con i giusti parametri:

```
$ route -n
$ netstat -r
```

l'opzione -n nel comando route serve per indicare che si deve scrivere solamente l'indirizzo IP delle righe della tabella. Infatti, se -n non è presente, il comando route prova a risolvere ogni indirizzo IP della tabella in nomi simbolici DNS (DNS reverse lookup). Se non è raggiungibile un server DNS, ciò comporterebbe un'inutile perdita di tempo.

Il "file" /proc/net/route non è un vero file. Perciò non è assolutamente il caso di modificarlo direttamente. E' meglio manipolarlo, per stabilire delle route statiche, solo attraverso il programma "route".

Opzioni importanti di route

add | del
Aggiunge (add) o toglie (del) una route dalla rete.

-net | -host
Indica che l'indirizzo che segue è una rete (-net) od un singolo host (-host).

Per aggiungere delle "strade" ad un tabella di routing si può usare il comando route, con l'opzione "add". Si può aggiungere un'intera rete, con l'opzione -net, o una singola stazione, con l'opzione -host.
Per togliere una voce da una tabella di routing si usa route con l'opzione "del".

Esempi:

```
# route add -net 192.168.13.0 netmask 255.255.255.128 dev eth0
```

Specifica che:
- si vuole aggiungere una route alla tabella (add)
- si tratta della strada che dovranno prendere tutti i datagrammi di una rete (-net)
- la rete di destinazione è la 192.168.13.0
- la netmask non è quella standard, ma 255.255.255.128 (bit più significativo dell'ultimo Byte a 1 nella maschera => la rete di classe C è divisa in due sottoreti; dato che l'ultimo Byte dell'indirizzo è 0, questa riga della tabella di routing tratta della sottorete più "bassa", con indirizzi IP da 192.168.13.1 a 192.168.13.126). Se la netmask fosse stata quella standard della classe dell'indirizzo della rete (nell'esempio 255.255.255.0), la si sarebbe potuta omettere nel comando route.

```
# route del -net 192.56.76.0 netmask 255.255.255.128 dev eth0
```

Elimina la stessa route inserita con il comando add precedente. Questo comando vuole tutti i parametri indicati, se se ne omette qualcuno, fidando sul fatto che la riga della tabella possa essere riconosciuta comunque, non funzionerà.
Al posto della maschera di sottorete si può usare un prefisso CIDR, separandolo dall'indirizzo con uno "slash". Il seguente comando fa lo stesso effetto del precedente "add":

```
# route add -net 192.168.13.0/25 dev eth0
```

Quando si vuole introdurre un router in una tabella di routing bisogna:
1. mettere una route che premetta di raggiungere in rete locale il router da usare, con il suo indirizzo IP.
2. mettere una route "di tipo gateway" all'indirizzo del router. Perché una route sia "di tipo gateway" bisogna crearla con l'opzione "gw"

Esempio:
```
# route add -net 192.0.0.07/20 dev eth1

# route del default netmask 0.0.0.0 dev eth1
```

Per introdurre un router di default, mettere l'opzione "default gw" prima dell'indirizzo del router; l'inserimento della maschera 0.0.0.0 può essere opzionale.
Esempio:
```
# route add default gw 36.243.112.1 netmask 0.0.0.0 dev eth1
```

Nella tabella di routing doveva essere sempre presente una riga che dirige al "localhost", instradando il traffico diretto a "questo" stesso computer. Di solito questa riga veniva aggiunta dal programma d'installazione della distribuzione. In tempi recenti modifiche al kernel hanno reso opzionale la presenza della la riga di localhost nella tabella di routing. Qualora fosse necessario, la route di localhost si introduce così:

```
# route add -host 127.0.0.1 dev lo
```

Abilitazione dell'"IP forwarding"

Di solito la funzione di trasferimento dei pacchetti IP da un'interfaccia ad un'altra ("IP forwarding"), cioè la funzione di routing, è disabilitata per sicurezza nell'installazione di default delle distribuzioni. Per questo la funzione va abilitata, scrivendo un 1 nel file "ip_forward", come indicato nell'esempio che segue:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Nota:
si noti che il programma route c'è anche in Windows, e funziona in modo simile. Per vedere la tabella di routing si deve usare l'opzione PRINT del comando route:
```
C:> route PRINT
```

1.12 NAT e firewall con iptables

Le funzioni di router NAT e di firewall di tipo packet filter sono incluse direttamente nel kernel di Linux e vengono chiamate "netfilter". Dalla versione 2.4 del kernel Linux il packet filter viene gestito con il programma "**IPtables**", che permette di emettere comandi di configurazione al packet filter.
IPtables non ha file di configurazione, per cui se si devono rendere persistenti le regole del firewall, cioè se si vuole che rimangano anche dopo la ripartenza del computer, è necessario fare uno script che imposti le regole ed occorre farlo partire con uno degli script .rc che vengono eseguiti all'accensione del computer[1].
IPtables lavora con diverse "tabelle", che gli fanno fare funzioni diverse. Le tabelle più significative sono "filter", che serve per le funzioni di packet filter, e "nat", per alterare gli indirizzi ed i port dei pacchetti ed ottenere funzioni di NAT/PAT.
iptables permette di applicare regole in momenti precisi del percorso del pacchetto all'interno del router. Il momento in cui si applica una regola viene detto "**chain**" ed ha possibilità diverse in base alla tabella utilizzata.
Per ogni pacchetto che il filtro tratta, esso esamina le regole ad una ad una e vede se esse corrispondono al pacchetto in transito.

- se il pacchetto non verifica le condizioni di una regola si passa alla successiva
- se la regola è verificata si esegue l'azione indicata dal "target", oppure si controlla un'altra regola, pure specificata attraverso il comando ipfilter
- se le regole finiscono si esegue quanto detto dalla regola di default (che viene chiamata "policy").

La sintassi generale di un comando iptables è la seguente:

```
# iptables [-t <tabella>] <comando> <chain> <regola> <target>
```

-t <tabella> indica il tipo di tabella, il default è la tabella "filter", che contiene le regole di filtraggio; un'altra tabella utilizzata è "nat", che server per il NAT/PAT. Esistono altri due tipi di tabella, che qui omettiamo.
<comando> è un comando in forma di opzione che indica a iptables cosa fare:

-L (list) elenca le regole esistenti, se si usa l'opzione –line-numbers mette anche i numeri delle regole
-A (append) aggiunge una regola in fondo alle altre
-I (insert) inserisce una regola in un punto stabilito, indicato con il suo numero DOPO l'indicazione della chain
-P (policy) cambia la politica di default della chain specificata
-F (flush) cancella tutte le regole, eventualmente anche per una sola chain, se viene specificata
-D (delete) cancella una regola, che si può esprimere come numero di regola oppure nello stesso modo usato per inserire nella tabella
-R (replace) sostituisce una regola

<**chain**> (punto di intervento della regola), nel seguito verrà definito anche "**catena**", nonostante non sembri termine molto adatto
se la tabella è di tipo "filter", può la "chain" può essere INPUT, FORWARD o OUTPUT.
se la tabella è di tipo "nat", può essere PREROUTING, OUTPUT o POSTROUTING.
<**regola**> (rule) è la regola, specificata come vedremo in seguito
<**target**> (obiettivo della regola) specifica cosa si deve fare quando la regola si applica al pacchetto che si esamina.
Nel comando iptables il target è preceduto da -j, che sta per "jump", in pratica si salta ad un'altra regola, che nella maggior parte dei casi è una regola terminale che fa "fare qualcosa", cioè un "target"
se la tabella è di tipo "filter", il target può essere uno di: ACCEPT, DROP o REJECT[2] (vedi)
se la tabella è "nat" alcuni dei target sono MASQUERADE, NETMAP, DNAT, SNAT, REDIRECT
Un target potrebbe anche rimandare ad un'altra regola ("jump")

Molte degli elementi dei comandi iptables invertono il loro significato se li si fanno precedere dal punto esclamativo.
iptables ha centinaia di opzioni che rendono molto flessibile e potente il suo uso.
E' possibile mettere delle regole basate sul tempo di ricezione dei pacchetti.
Con iptables si possono definire ed usare delle variabili, come le variabili d'ambiente di un normale script di shell.
Per definirle:
VARIABILE="stringa"
Per usarle:
$VARIABILE

1.12.1 NAT con iptables

Tramite le funzioni NAT di iptables è possibile fare in modo che indirizzi IP pubblici corrispondano ad indirizzi IP privati.
La tabella "nat" di ipfilter ha tre "chain":

1 Per esempio, in Fedora, in /etc/rc.d/local.rc, script che viene sempre eseguito alla fine di tutte le inizializzazioni effettuate dagli altri script "rc".
2 Ci sono anche altri "target" qui trascurati.

- PREROUTING
  che serve a modificare i pacchetti non appena arrivano al router, il routing deve essere fatto con il "nuovo" indirizzo
- FORWARD
  che altera i pacchetti mentre vengono spostati da un'interfaccia ad un'altra
- POSTROUTING
  che modifica i pacchetti un momento prima di spedirli sull'interfaccia di destinazione dopo che il routing è già stato fatto con il vecchio indirizzo

NAT "statico" (DNAT e SNAT)

Se si vogliono sostituire "staticamente" indirizzi IP interni ad indirizzi IP esterni si possono usare i target SNAT, e DNAT. Con questi target si possono far cambiare anche i port, ma non è indispensabile; non specificando alcun port nelle opzioni, il router tenderà a mantenere i port originali.

SNAT (Source NAT, NAT "classico")

Viene alterato l'indirizzo IP del sorgente (mittente) e sostituito con quello configurato. Si applica solo alla catena di POSTROUTING.
SNAT serve per cambiare uno o più indirizzi privati interni con uno o più indirizzi pubblici statici esterni,

L'opzione di questo target è:
--to <IndirizzoIP o Range di indirizzi IP "sorgente", che sostituiscono l'originale>

Esempio (Source NAT):

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 33.21.4.7
```

Tutti i pacchetti che escono dall'interfaccia eth0 prendono l'indirizzo 33.21.4.7. Questo è il caso in cui ci sia stato assegnato l'indirizzo IP statico 33.21.4.7 e vogliamo usare su Internet tutti i computer attaccati al nostro router NAT con l'unico indirizzo con cui possiamo andare in Internet.

Significato delle opzioni:

-A	comando: A = aggiunge la regola alle altre
-t nat	tabella: usa la tabella nat
POSTROUTING	chain: indica che la regola viene applicata sul pacchetto che esce, dopo il routing. Il pacchetto modificato seguirà la strada indicata dall'indirizzo di destinazione DOPO la sostituzione
-o eth0	interfaccia di uscita: indica che la regola si applica ai pacchetti che escono attraverso quella scheda di rete
-j SNAT	il target SNAT: indica il "source NAT"
--to <IP>	opzione del target SNAT, indica l'indirizzo FINALE (dopo la sostituzione).

DNAT (Destination NAT)

Il target DNAT indica che viene alterato l'indirizzo IP di destinazione (del destinatario) e sostituito con quello configurato con il comando iptables. Si applica normalmente alla chain di PREROUTING[1], dunque interviene sul pacchetto prima che si decida dove mandarlo.
In questo caso il router sostituisce ogni indirizzo interno con il suo corrispondente esterno. Questo tipo di target è adatto per quando si hanno uno o più IP pubblici "esterni" che si vogliono rimappare su IP privati "interni", per esempio una "server farm" che vuole usare un solo indirizzo IP ma suddividere su più server fisicamente diversi il lavoro sui diversi protocolli (un server Web avrà un indirizzo interno diverso da un server FTP, ma entrambi avranno lo stesso indirizzo esterno).

L'opzione di questo target è:
--to <IndirizzoIP o Range di indirizzi IP "destinazione", che sostituiscono l'originale>

Esempio ("destination NAT"):

```
iptables -t nat -A PREROUTING -p tcp -i eth0 --dport 80 -j DNAT --to 192.168.13.128:8080
```

Tutti i pacchetti TCP in ingresso sulla eth0 e che sono rivolti al port 80 sono mandati (destination!) al port 80 della macchina 192.168.13.128. E' la situazione in cui tutto il traffico HTTP rivolto in ingresso alla nostra rete viene mandato ad uno specifico server interno.

Le nuove opzioni usate significano:

1 Anche a OUTPUT es alle catene definite dall'utente

PREROUTING chain: indica che la regola viene applicata sul pacchetto "originale", prima del routing.
Il pacchetto modificato seguirà la strada indicata dall'indirizzo di destinazione DOPO la sostituzione
-p tcp protocollo: sono interessati dalla regola i datagrammi che trasportano segmenti TCP
--dport <port> port di destinazione: indica il port di destinazione ORIGINALE prima delle sostituzioni
-j DNAT il target DNAT: indica il "destination NAT"
--to <IP>:<port> opzione del target DNAT, indica l'indirizzo ed il port FINALE (dopo la sostituzione)

Esempio: ("**port forwarding**")

```
iptables -t nat -A PREROUTING -p tcp -d 33.21.4.7 --dport 80 -j DNAT --to 192.168.13.128:8080
```

Tutti i pacchetti che hanno l'indirizzo di destinazione 33.21.4.7 ed il port remoto 80 vengono modificati introducendo come indirizzo di destinazione 192.168.13.128 e come port remoto 8080. Questa regola fa raggiungere un determinato server (verosimilmente HTTP, sul port 8080) sulla rete interna quando gli vengono rivolte richieste dall'esterno su un altro indirizzo 33.21.4.7 ed un altro numero di port.

Le nuove opzioni usate significano:

-d <IP> destinazione: l'indirizzo destinazione ORIGINALE

NAT "masquerading" con iptables

Il "mascheramento" di un indirizzo si utilizza quando si ha a disposizione un unico indirizzo IP pubblico dinamico e si vuole permettere a più stazioni di una rete locale di comunicare con un'internetwork esterna. Gli indirizzi IP privati vengono "nascosti" all'internet esterna e sostituiti in ogni datagramma con l'unico indirizzo IP pubblico.
Il "mascheramento" avviene dopo il routing ed è adatto a sostituire gli indirizzi privati iniziali con indirizzi pubblici dinamici. Infatti il target MASQUERADE è del tutto simile ad un SNAT, con la sola differenza che usa come indirizzo esterno non un IP fisso, ma l'IP che è attualmente assegnato all'interfaccia indicata nel comando. Ciò significa che ogni volta che la regola si applica, il software deve andare a vedere qual è l'attuale indirizzo dell'interfaccia di uscita, cosa che costa un po' di tempo.

Vediamo un esempio per configurare il packet filter con iptables in modo da fargli effettuare un masquerading.
Supponiamo che:

- l'indirizzo pubblico sia, per esempio, 39.223.116.111/21 (prefisso corrispondente alla netmask 255.255.248.0), e sia raggiungibile tramite l'interfaccia eth1 (eventualmente collegata ad un router ADSL).
  L'indirizzo di rete che corrisponde a questo indirizzo pubblico e netmask è: 39.223.112.0
- il default gateway sia 39.223.116.1/21
  raggiungibile tramite eth1
- gli indirizzi privati siano quelli della rete 192.168.13.0/24
  raggiungibili tramite eth0

La tabella di routing dovrà essere configurata in modo che risulti fatta così (usando ifconfig per configurare le due interfacce, con il relativo prefisso, dovrebbero essere aggiunte automaticamente le due prime righe, mentre si potrebbe essere costretti ad aggiungere manualmente la terza):

```
# route
Destination     Gateway         Genmask         Flags   Metric  Ref Use  Iface
192.168.13.0    *               255.255.255.0   U       0       0   0    eth0
39.223.112.0    *               255.255.248.0   U       0       0   0    eth1
default         39.223.116.1    0.0.0.0         UG      0       0   0    eth1
```

Per aggiungere eventualmente l'ultima riga:

```
# route add default gw 39.223.116.1 netmask 0.0.0.0 dev eth1
```

Facendo attenzione a non scordarsi di abilitare il routing, come visto in precedenza, si deve ora configurare il packet filter per il masquerading; ciò viene fatto con il seguente comando[1]:

```
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Il comando imposta la tabella nat (-t nat NON è opzionale!) per aggiungere una riga nella "chain" POSTROUTING (il router modifica i datagramma che stanno uscendo) sull'interfaccia di uscita (-o) eth1 e con il target MASQUERADE, che vuol dire che si deve utilizzare un solo indirizzo pubblico di uscita. Si noti che il comando iptables appena illustrato non specifica alcun indirizzo IP; infatti questa configurazione è valida anche nel caso in cui l'indirizzo IP pubblico sia dinamico e configurato tramite DHCP[2].

1 Se invece di un modem con collegamento Ethernet si avesse a disposizione un modem USB o seriale, invece di un'interfaccia "eth" si sarebbe verosimilmente utilizzata una "ppp".

```
# iptables -t nat -L POSTROUTING
```

Visualizzando la configurazione attuale del NAT di questo sistema:

```
# iptables -t nat -L POSTROUTING
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE all  --  anywhere             anywhere

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
```

Dove si vede che l'unica "catena"[1] configurata è quella di POSTROUTING e non ci sono limitazioni né ai protocolli né agli indirizzi; dunque non c'è di fatto filtraggio, ma solo NAT/PAT su un solo indirizzo "esterno".

Target NETMAP
Con il target NETMAP la parte di host degli indirizzi che vengono sostituiti non viene cambiata. Con questo target è possibile fare una corrispondenza 1:1 fra gli indirizzi interni ed esterni, pur sostituendo la parte di rete degli indirizzi. I bit a uno della maschera saranno sostituiti, i bit a zero saranno lasciati uguali.

Esempio:

```
iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j NETMAP --to 33.21.4.0/24
```

Questa regola cambia tutti gli indirizzi IP della rete interna 192.168.1.0/24 nei corrispondenti indirizzi della rete 33.21.4.7/24. Se p.es. avessimo l'indirizzo 192.168.1.7 esso diventa 33.21.4.7.

Le opzioni usate significano:
-s <IP> source: l'indirizzo sorgente
-j NETMAP il target DNAT: indica il "destination NAT"
--to <IP> opzione del target DNAT, indica l'indirizzo ed il port FINALE (dopo la sostituzione)

Target REDIRECT
Il target REDIRECT cambia l'indirizzo di destinazione in modo tale da mandare i pacchetti alla stessa macchina ove è il router (localhost). Questo può servire per fare un proxi "trasparente" ove tutti il pacchetti che vengono mandati al port 80 del default router vengono in realtà spediti al proxi HTTP che risiede su quella stessa macchina.
Esempio:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

Tutti i pacchetti che passano di qui e sono diretti a qualsiasi indirizzo ed al port 80, sono passati al port 8080 di QUESTA macchina.

1.12.2 Firewall con iptables

Il packet filter di Linux utilizza la tabella "filter", che è quella di default del comando iptables. Dunque se nel comando iptables non si specifica l'opzione "-t", è come specificare "-t filter".
Il firewall basa le sue "decisioni di filtraggio" su tre "chain" su ciascuna delle quali possono essere specificate delle regole. Le tre chain sono relative al "momento di intervento", cioè alla "posizione" nella quale la regola viene applicata.
I momenti di intervento in cui si applicano le regole sono:

- INPUT: traffico che è destinato al firewall
  è il traffico dei datagrammi IP che hanno come indirizzo di destinazione quello di una delle interfacce del firewall router. Possono provenire sia dall'"esterno" (rete Internet") che dall'interno (rete privata) o dallo stesso computer (destinazione 127.0.0.1) e da qualsiasi interfaccia. E' il punto in cui è più indicato effettuare i filtraggi
- FORWARD: traffico che è inoltrato dal firewall
  è il traffico che viene smistato dal firewall, da una sua interfaccia ad un'altra. Anche qui è opportuno filtrare
- OUTPUT: traffico che parte dal firewall
  è il traffico che esce dal firewall, a partire dal computer su cui gira il netfilter. I datagrammi di questo traffico hanno come indirizzo del mittente quello di una delle interfacce del firewall router. Di solito è meglio filtrare

2 Si noti che se si usa il DHCP la tabella di routing viene configurata automaticamente nella riga dell'interfaccia di accesso alla internet e nella riga del default gateway
1 Vedi oltre per dettagli sulle "catene"

prima dell'uscita, se non altro perchè se si filtra all'uscita come minimo si fa fare del lavoro per nulla al computer, che prepara dei pacchetti che poi potrebbero essere scartati.

Un pacchetto "passa" da una sola di queste "chain": se proviene dall'"esterno" passa da INPUT, se viene instradato passa da FORWARD, se viene originato all'interno del firewall passa da OUTPUT.
Nei momenti di intervento il packet filter usa le regole nell'ordine con cui sono memorizzate e quando trova una corrispondenza fra una regola ed il pacchetto, applica ciò che viene specificato dall'obiettivo della regola (target).
Le regole hanno un obiettivo, che, quando la tabella è di tipo "filter" può essere: ACCEPT, DROP o REJECT[1]. Accept fa passare il datagramma, Drop lo blocca senza darne comunicazione al mittente, Reject blocca il pacchetto e ne dà comunicazione al mittente.
Il firewall controlla le regole nell'ordine con cui sono impostate tramite iptables, per cui la successione dei comandi iptables che si usa può essere importante.
Per questioni di efficienza bisognerebbe fare in modo che le regole che hanno la maggiore probabilità di essere applicate vengano prima nell'ordine di scansione del firewall.

Esempi:

```
# iptables -F
```

cancella tutte le regole del firewall, tranne le politiche di default. Per cambiare le politiche di default, usaare -P. Se dopo -F si fosse indicato un punto di intervento od una catena, sarebbero state cancellate tutte le regole solo di quell'elemento.
Di solito in ingresso è bene non accettare nulla, tranne quanto esplicitamente richiesto dall'interno, per questo si possono mettere le seguenti policy di default:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT ACCEPT
```

con i comandi appena visualizzati, sono implementate queste policy (-P) di default:

- INPUT DROP rifiuta assolutamente ogni ingresso "autonomo" dalla rete esterna
- FORWARD DROP non fa il routing
- OUTPUT ACCEPT concede l'accesso all'esterno da parte degli host interni

Dopo aver stabilito queste rigide regole di default, le si faranno precedere da altre regole, che apriranno l'accesso solo ai servizi effettivamente necessari.
L'ultima regola specificata è un po' controversa; alcuni l'accettano, ma c'è da considerare il fatto che l'obiettivo di molti attacchi di malintenzionati è la presa di controllo dei computer nelle reti degli altri[2]. Se qualcuno prende il controllo di un computer all'interno della nostra rete, non è bello lasciargli libera la possibilità di uscire dalla rete con il computer "zombizzato". Dunque, anche se ciò comporta più noie per l'amministratore della rete, l'ultima regola andrebbe riscritta, in modo un po' più paranoico, così:

```
# iptables -P OUTPUT DROP
```

Le regole più comuni

Liberare localhost

Se adottiamo le politiche, sagge, di bloccare tutto e poi di liberare solo quello che serve, la prima cosa che ci serve liberare è l'accesso al nostro stesso computer (localhost) che viene anch'esso bloccato.
Per poter contattare il nostro stesso sito Web o per fare ping sulle nostre stesse interfacce, bisogna sbloccare l'accesso in ingresso ed in uscita a tutti i datagrammi che hanno indirizzo IP 127.0.0.1:

```
# iptables –t filter -A INPUT -s 127.0.0.1 -j ACCEPT
# iptables –t filter -A OUTPUT -d 127.0.0.1 -j ACCEPT
```

Come già detto, l'indicazione "–t filter" è opzionale. La prima delle due regole precedenti fa passare tutti i datagrammi che entrano nel computer ed hanno "localhost" (127.0.0.1) come indirizzo del mittente (sorgente (-s)). La seconda lascia uscire tutti i datagrammi, indipendentemente dal protocollo che trasportano, che hanno come indirizzo di destinazione (-d) "localhost".
In modo analogo si possono specificare gli indirizzi IP singoli in tutte le altre regole di iptables.
E' anche possibile filtrare indirizzi MAC, basta usare l'opzione "mac-source":

--mac-source [!] <indirizzo MAC>

L'indirizzo MAC va scritto come al solito (AA:BB:..).

Filtraggio ICMP

Al giorno d'oggi non è una buona idea lasciar passare liberamente tutti i datagrammi ICMP, dato che possono essere usati per molti tipi di abuso. E' peraltro utile per scopi diagnostici avere la possibilità di emettere, e qualche volta anche

1 In verità l'obbiettivo potrebbe anche essere il salto (jump) ad una catena "custom", definita dall'utente, fatta di altre regole, ma le cose si complicherebbero troppo ..
2 In questi casi si dice che l'attaccante ha reso il nostro computer uno "zombie" a sua disposizione.

di ricevere, comandi "ping". Dato che ping usa frame ICMP su datagrammi IP , è possibile filtrare i datagrammi basandosi sul contenuto del frame ICMP,
Riviste e siti Web pubblicano script di varia natura che mostrano come gli esperti di sicurezza configurano il filtraggio degli ICMP.

Abilitare "blocchi" di indirizzi IP
Usando una coppia indirizzo / prefisso è possibile filtrare i blocchi di indirizzi IP che corrisponderebbero alla rete individuata dalla maschera.

```
# iptables -A INPUT -s 172.16.0.40/29 -p tcp --dport 80 -j ACCEPT
```

La precedente regola stabilisce che possono entrare nel firewall solo pacchetti "http" che provengono dagli indirizzi IP che vanno da 172.16.0.40 a 172.16.0.47. Infatti questa "rete" avrebbe 3 bit di parte di host (prefisso 29), l'indirizzo di rete sarebbe 172.16.0.40, gli indirizzi ..0.40 (indirizzo di rete) e ..0.47 (indirizzo di broadcast) non dovrebbero essere usati se questa fosse una "vera" sottorete, in verità questo mascheramento serve solo per sapere se il pacchetto deve essere filtrato o meno, per cui anche i datagrammi provenienti dagli indirizzi 172.16.0.40 e 172.16.0.47 sono legittimi ed entrano nel sistema.
Un altro modo per specificare un blocco di indirizzi è attraverso l'opzione "range":
[!]--src-range <indirizzo IP>-<indirizzo IP>
[!]--dst-range <indirizzo IP>-<indirizzo IP>

Filtraggio di http e di altri protocolli di livello trasporto

Le opzioni "-p tcp" e "-p udp" hanno tante "sottoopzioni" che permettono di filtrare i segmenti in base al contenuto di dei campi, in particolare flag e opzioni, che sono i campi più pericolosi dal punto di vista degli attacchi.
Per sapere rapidamente le opzioni che si possono utilizzare con i vari protocolli si può usare # iptables -p <protocollo> -h.

Uscita http su proxi
Uscita http solo su un proxi server di indirizzo ben definito, su port 8080. I dati di ritorno tornano solo da quel proxi.

```
# iptables -A OUTPUT -d 172.16.0.254 -p tcp --dport 8080 -j ACCEPT
# iptables -A INPUT -s 172.16.0.254 -p tcp --sport 8080 -j ACCEPT
```

Invece del numero della porta si può usare il nome del servizio, così come specificato nel file /etc/services. Inoltre è possibile usare un nome di host in luogo dell' indirizzo IP.
Se nella regola si vuole mettere una lista di port invece che un singolo port, si può usare l'opzione "source-ports":
--source-ports <lista di port separati da virgole>
--destination-ports <lista di port separati da virgole>
In luogo di questi comandi si può anche usare --sports e --dports

1.12.3 Programma traceroute

1.12.4 Programma finger (port 79)

Permette di vedere quali sono gli utenti correntemente collegati come terminali al computer. Può essere accessibile anche da rete IP, se abilitato (di solito non lo è).
Esempi:

```
# finger root@BrontoloLinux
Login: root Name:root Directory: /root Shell: /bin/bash
On since Thu Apr 26 22:36 (CEST) on tty1 2 hours 15 minutes
idle
New mail received Thu Apr 26 22:37 2001 (CEST)
Unread since Sat Apr 21 11:30 2001 (CEST)
No Plan.
```

1.12.5 Server e client telnet (port 23)

telnet è un programma di collegamento con terminale remoto, che comunica con il computer centrale attraverso una qualsiasi rete TCP/IP. Si può lanciare specificando l'host od il suo indirizzi IP, oppure senza indirizzo. In quest'ultimo caso si potrà iniziare un collegamento telnet con un host aprendo la comunicazione dal prompt telnet:

```
telnet> open <Nome o IP dell'host cui mi voglio collegare>
```

Per chiudere un collegamento:

```
telnet> close
```

Durante la sessione di collegamento è possibile richiedere al server telnet l'elenco dei comandi disponibili digitando help.
Dato che pone problemi riguardo alla sicurezza, il server telnet (demone telnetd) nelle distribuzioni odierne non viene di solito abilitato per default, alcune volte non viene nemmeno installato e bisogna specificarlo esplicitamente se si vuole che venga copiato dal CD al computer in fase di installazione.
Per default telnet non accetta login di root da remoto. Infatti il login di root è possibile solo da quei terminali che sono compresi nell'elenco del file /etc/securetty, che per default comprende solo le console virtuali che sono fisicamente sul computer centrale su cui è installato il S.O.
Peraltro una volta che ci si è connessi al computer usando un nome d'utente qualunque è poi possibile, conoscendo la password di root, ottenerne quasi tutti i privilegi digitando il comando "su" (super user) da telnet.

1.12.6 iptraf

Programma di monitoraggio delle prestazioni che funziona con interfaccia a carattere.

1.13 Server DHCP

Il server DHCP deve usare interfacce "reali": non può essere configurato su un alias IP.

Nelle distribuzioni RedHat – Fedora c'è uno script "rc" che lancia il server DHCP, è "**/etc/rc.d/init.d/dhcpd**" e viene lanciato dai vari script "rc" al boot ed ai cambi di runlevel. Lo si può usare anche "a mano" od in script dell'amministratore, nel modo "standard" per lanciare i servizi:

```
# /etc/rc.d/init.d/dhcpd start
     oppure
# /etc/rc.d/init.d/dhcpd stop
```

1.14 NFS (port 2049, UDP)

Network **F**ile **S**ystem è un protocollo definito dalla Sun per le sue stazioni diskless ed ora adottato in tutto il mondo Unix.
Fa uso, per i suoi servizi di rete di più basso livello, del protocollo RPC (**R**emote **P**rocedure **C**all), anch'esso sviluppato da Sun.

Configurazione del client NFS

Se i demoni che servono per il funzionamento di NFS sono accessibili dal kernel e sono lanciati al boot di sistema per accedere ad un directory condiviso NFS basta montare un filesystem di tipo NFS:

```
mount <NFSserver>:</directory/esportata/da/NFSserver/> </directory/ove/si/monta>
```

<NFSserver> è un nome di dominio DNS, un nome presente nella tabella /etc/hosts oppure un indirizzo IP.

Se non c'è un DNS attivo, oppure una tabella /etc/hosts sul computer client, si possono usare solo indirizzi IP (non c'è "browsing" automatico dei nomi come in NetBIOS).
Il mount è del tutto "normale", identico a quello che si farebbe con un filesystem locale e si può anche mettere in fstab per la connessione diretta in fase di bootstrap. Naturalmente il filesystem nfs si può "smontare" con umount.

Configurazione del server NFS

Perchè NFS funzioni, è necessario che il "portmapper" sia in funzione. Questo programma viene usato da ogni programma RPC per mantenere l'associazione dinamica fra il programma ed il numero port che utilizza. Trasforma le richieste ai numeri di programma RPC in numeri di port TCP o UDP. Il demone "portmapper" si può chiamare portmap o rpc.portmap. Per verificare che stia girando si può usare ps, per esempio così:

```
root$ ps aux | grep portmap
```

Altri due demoni necessari per NFS sono mountd e nfsd.
Le distribuzioni scrivono automaticamente nei vari file rc di boot, in modo che, se si vuole installare il server NFS, i tre demoni siano lanciati automaticamente al boot.
Per far condividere con NFS una directory bisogna "esportarla", scrivendola nel file /etc/exports. In questo file si definiscono le directory che possono essere usate dagli utenti di rete (gli "share" in gergo Windows).
Ogni riga di exports definisce un directory da esportare ed è composta dalla path di quel directory e da una lista di host IP che possono accedervi. Inoltre possono essere presenti alcune opzioni che specificano come deve essere fatto il mount.

Il formato di una riga è il seguente:

<PathInizio> [<hostname o indirizzoIP/netmask>] [**(**opzioni**)**] ..

<PathInizio> è il directory di "export", che si deve condividere

<hostname> è il numero IP od il nome IP dell'host. Può essere abbreviato con * e ?, si può usare @ per i gruppi, <hostname> è un nome di host in rete, non di utente!
Il formato del file exports è spiegato in dettaglio nella sua pagina man.
Con il comando showmount (opzione -e ("exports")) da lato client si possono chiedere i filesystem esportati dal server, indicando il nome o l'indirizzo del server remoto:

```
$ showmount -e <hostname o indirizzoIP>
```

Se invece si lancia showmount senza opzioni, fa vedere gli utenti che nel corso del tempo (anche nel passato) hanno montato il directory condiviso.

```
$ showmount
```

Vediamo ora un esempio di /etc/exports :

```
/ boss (rw) vice (ro) # tutto il filesystem dato solo ai computer più importanti
# vice ha solo diritto di lettura
/usr/monti *.monti (rw) # directory condiviso fra tutti gli utenti del dominio DNS monti
/usr/UffTecNFS @UfficioTecnico (rw) # tutti gli utenti che appartengono al gruppo NIS
"UfficioTecnico"
/home/gamon gamon*.* (rw) # tutti i computer il cui nome contiene gamon come dominio
"primario"
# es. gamon386.monti, gamonPent.monti
/mnt/cdrom 192.168.0.255/255.255.255.0 (ro,insecure) # tutti gli utenti della rete loca-
le, insecure significa che non c'è # limitazione sul numero di port TCP su cui provengono
le richieste (2049), che normalmente è quello # riservato dagli standard
/mnt/floppy 192.168.0.255/255.255.255.128 (ro,insecure) # una sottorete della rete locale

/mnt/master 192.168.0.4 (ro,insecure) # la macchina vicina (fisicamente)
```

La tabella /etc/exports è molto sensibile a piccoli "errori" di sintassi. Bisogna provare con pazienza, cercando di individuare in modo "esatto" quali sono i computer che possono avere accesso. Ciò soprattutto per questioni di sicurezza.

Dopo aver modificato /etc/exports è necessario fermare e far ripartire i demoni rpc.nfsd e rpc.mountd, per esempio in questo modo:

```
root$ killall -HUP /usr/sbin/rpc.nfsd
root$ killall -HUP /usr/sbin/rpc.mountd
```

oppure rifare il boot del sistema.

1.14.1 SMB (port 137, 138 e 139)

SMB (Service Message Blocks) | CIFS (Common Internet File System) è il protocollo di condivisione di risorse utilizzato nelle reti Microsoft. Esso è un'estensione dei servizi di NetBIOS, protocollo utilizzato nella prima rete locale IBM-Microsoft (LanManager) , e, dopo la "separazione" fra IBM e Microsoft, portato avanti nelle reti locali Microsoft con il nome di NetBEUI (NetBIOS extentions), e divenne il protocollo Microsoft principale di basso livello fino a Windows 2000 server. Nelle reti Microsoft attuali il protocollo di basso livello utilizzato è sempre IP, per cui i servizi di rete Microsoft (SMB, CIFS), che pure esistono ancora ed anzi sono espansi e modificati "quotidianamente", funzionano venendo incapsulati in datagrammi IP ("NetBIOS over IP") e si può fare a meno di NetBEUI.
Oltre che IP e NetBEUI, SMB può usare i servizi di basso livello anche di IPX/SPX (Novell Netware) . Con SMB (il cui nome oggi viene fatto coincidere con NetBIOS), è possibile condividere directory e stampanti. Esso viene utilizzato nei sistemi Windows, sia Win9X che Win NT-XP.
Linux è in grado di utilizzare il protocollo SMB, emulando perciò un client od un server Windows. Per utilizzare SMB in Linux si deve usare la suite di programmi chiamata "Samba" (progetto iniziato e tuttora guidato da Andrew Tridgell, Camberra, Australia).
Samba funziona **solo** con il TCP/IP (NETBIOS over IP) non su NetBEUI o IPX.

Configurazione del client Samba

Samba mount permette il mount di share di rete Windows in un filesystem Linux.
Sintassi:

mount -t smb <nome della risorsa condivisa NetBIOS> <mountpoint>

<nome della risorsa condivisa NetBIOS> appare come una path che inizia con due slash invece di uno. è un nome "di share" Windows, che si compone di:

//<Nome NetBIOS del server>/<path e Nome dello "share">

Il nome del server lo si esprime facendolo precedere da due slash, che possono essere "dritti", alla moda Unix, o retroversi (back slash), alla Windows. L'indirizzo del computer che corrisponde al nome NetBIOS può essere trovato dal computer Linux in vari modi, cercando in lmhosts, hosts, DNS, wins, o con una richiesta in broadcast a tutti i computer accesi attualmente sulla rete locale[1]. L'ordine indicato corrisponde a quello usato per default da Samba; se è necessario esso si può anche modificare.
Inoltre bisogna rimarcare che i nomi di dominio TCP/IP ed i nomi NetBIOS non sono necessariamente uguali; con Samba si può usare un nome per farsi vedere da Windows ed un altro nome per il DNS TCP/IP.
<Nome NetBIOS del server> può anche essere un indirizzo IP.
L'utente può montare il directory solo se gli viene assegnato il diritto, per cui al tentativo di mount viene richiesta una password. Se lo share che si vuole montare non ha password, basta premere Enter alla richiesta di password.

In alternativa a mount con opzione -t smb si può usare il comando smbmount:

smbmount <nome della risorsa condivisa NetBIOS> <mountpoint> **-o** <opzioni>

La più importante delle <opzioni> è username=<nome utente SMB nel computer cui ci si collega>
Questa opzione è utile se si è fatto il login su Unix con un nome di utente diverso da quello che serve per collegarsi alla risorsa SMB desiderata.

Esempio:
```
# smbmount //goku/gokuC /rete -o username=gamon
```

Monta su /rete la condivisione Windows di nome gokuC sul computer di nome goku. Invece del nome d'utente Unix utilizza lo username "gamon".

Esiste anche un "Samba client", che permette un collegamento ad un server Win, con un client simile a quello di FTP, con il quale si può fare il browsing di file e la stampa.

```
# smbclient '\\ServerNetBIOS\NomeShare'
```

collega allo share NetBIOS NomeShare, sul computer di nome ServerNetBIOS. Il collegamento avviene dopo che smbclient ha chiesto una password.

```
# smbclient '\\ServerNetBIOS\NomeStampantè -P
```

collega alla stampante condivisa NomeStampante, collegata al computer di nome ServerNetBIOS.

Il collegamento avviene dopo che smbclient ha chiesto la password per la risorsa ricercata.

```
# smbclient -L <nomeNetBIOS>
```

visualizza l'elenco dei servizi SMB che <nomeNetBIOS> mette a disposizione, dopo averne chiesto la password.

Il file di configurazione di samba è /etc/samba/smb.conf

I daemon di Samba sono smbd e nmbd, la prima cosa da controllare se Samba non funziona è se questi demoni sono lanciati (es. # ps aux | grep smbd).

/etc/samba/lmhosts

Questo file è analogo al file /etc/ hosts, solo che funziona per i servizi NetBIOS, non a caso sta nel sottodirectory samba.

Questo file è composto di linee di testo, ciascuna delle quali definisce un indirizzo. Il formato delle linee è:

<Indirizzo IP> <Nome NetBIOS>

Analogo a lmhosts, in Windows 9X c'è C:\windows\lmhosts.sam, in Windows NT-XP
C:\WINNT\system32\drivers\etc\lmhosts.sam

1.15 File di configurazione di Samba, per client e server

I file di configurazione di samba sono in /etc/samba (distr. Fedora). Il "primo" file di configurazione è smb.conf.

Un esempio commentato di smb.conf

```
[global]
workgroup = MONTI
server string = Samba Server (Pentium 100 32M)
guest account = nobody
keep alive = 30
```

1 Quest'ultima cosa spesso non funziona!

```
os level = 2
security = share
; share = protezione con password della singola risorsa (tipo Win 98)
; user = protezione con autenticazione dell'utente su QUESTO computer
;       usa un insieme di utenti specifico di samba, non gli utenti Unix
;       ogni utente samba per i suoi diritti Unix deve avere una corrispondenza
;       con un utente Unix
; server = protezione con autenticazione dell'utente presso un "password server" esterno
;          naturalmente gli utenti da autenticare vanno creati su questo computer!
; domain = protezione con autenticazione dell'utente presso un "controllore di dominio"
;          di sicurezza Win NT (può anche essere un computer Samba)
; Se si vuole autenticare gli utenti presso un sistema Windows NT si deve fare così:
; Esempio per accesso tramite password server:
; security = server
; password server = 192.168.1.10
; encrypt passwords = yes

; modalità' di utilizzazione delle stampanti "lato Unix":
printing = bsd
printcap name = /etc/printcap
load printers = yes
wins support = no
; definizione dell'accesso da parte degli host (indirizzi IP)
hosts deny = * ; per default nessun computer puo' accedere

;   permetto l'accesso ai computer della rete "1" tranne una sua sottorete da 64 stazioni
;   (6 bit di indirizzo host) che ha 01XXXXXX nell'ultimo byte di indirizzo.
; inoltre permetto l'accesso all'host di nome Paperino ed a tutti gli utenti del gruppo
; Linux GruppoSicuro, tranne (deny) l'utente NonMiFido:

hosts allow = 192.168.1. except 192.168.1.64/255.255.255.192,Paperino,@GruppoSicuro deny
= NonMiFido@GruppoSicuro

; I seguenti comandi permettono di far agire il nostro computer come
; login server per il client Windows 95/98 clients
; logon script =login%U.bat ; lo script che viene lanciato è
; login"nome dell'utente".bat

; logon script =login%m.bat ; cosi' invece login"nome della macchina".bat
; domain logons = yes
; domain master = yes

; QUI SEGUONO LE DEFINIZIONI DEGLI "SHARES"
; [netlogon]
; path = /netlogon
;[homes]
; comment = Home Directories
; browseable = no
; writable = yes
; create mode = 0750 ; maschera ottale "owner group other"
; da assegnare ai file creati "da rete"
# script Unix che viene lanciato quando si accede a questo share
SetupCommand = ScriptPartenza
; Accesso al CD per tutti gli utenti:
[cdrom]
path = /mnt/cdrom
comment = CD-ROM
volume = "Etichetta CD-ROM"
read only = yes
; locking = no
available = yes
share modes = no
browseable = yes
public = yes
; le seguenti per condividere le stampanti:
[printers]
comment = Le stampanti de computer Linux
browseable = no
printable = yes
public = no
read only = yes
create mode = 0700
directory = /tmp
; una "ardita" condivisione di tutto il filesystem
```

```
; !! NON FATELA !!
[root]
path = /
comment = Tutti i dischi di BrontoloLinux
read only = no
browseable = yes
public = yes
create mode = 0750
; read only = no
; guest only = no
; writable = yes
; only users = no
; admin users = gamon, root
; dont descend = /proc, /dev
```

Se il server Samba ha più di una interfaccia ethernet, è meglio specificare a quale interfaccia Samba si deve "attaccare". Il modo per farlo è scrivere, nella sezione global di smb.ini: [global] interfaces = / Esempio: `interfaces = 192.168.1.1/24` Il prefisso 24 è quello giusto per una normale rete di classe C, si dovrà cambiare se si fa un subnetting sulla rete. Per verificare il funzionamento dei file di configurazione di samba usare:

```
$ testparm
```

se samba interpreta bene il file di configurazione, il programma testparm risponde con "OK".

File /etc/samba/smbusers

Un esempio di file smbusers:

```
# Unix_name = SMB_name1 SMB_name2 ...
root = administrator admin
nobody = guest pcguest smbguest
```

Per gestire gli utenti e le password c'è anche il comando di console smbpasswd.
Se usato da un utente normale smbpasswd serve solo per cambiare la password di quell'utente. Se usato da root permette di cambiare le password di tutti gli utenti ed anche di aggiungere nuovi utenti.
Opzioni più interessanti di smbpasswd:
-U <username> permette di cambiare la password dell'utente specificato.
-r <Nome della macchina remota> permette di cambiare la password sulla macchina remota specificata (la macchina NON può essere Win 9X)
-a <username> ordina di introdurre un nuovo utente con lo username indicato, sulla macchina locale.
Se non esiste un utente Unix con lo stesso nome dell'utente Samba che si vuole creare, bisogna prima creare l'utente Unix, p.es. con adduser.
Esempio:

```
# adduser nuovo
# smbpasswd -a nuovo
```

-n (senza parametri) memorizza una password vuota per l'utente specificato con altre opzioni

Per default Samba NON accetta password vuote; se si vuole che vengano accettate, si può mettere la riga
`null passwords = yes`
nella sezione [global] di smb.conf

Il comando di shell smbstatus mostra tutte le connessioni SMB correnti.

SWAT

samba può anche essere gestito in remoto via http. Il servizio che si deve usare è SWAT. le distribuzioni più recenti ne possono fare l'installazione. Deve essere presente la linea:

```
swat 901/tcp
```

nel file /etc/services (901 è il numero del port TCP cui ci si collega) ed anche

```
swat stream tcp nowait.400 root /<path di swat> swat
```

nel file /etc/inetd.conf, oppure una configurazione equivalente nei file di configurazione di xinetd.
Se per caso si devono aggiungere queste linee, poi è necessario far interrompere e riprendere il demone internet (inetd) in questo modo:

```
killall -HUP inetd
```

Per usarlo aprire un browser sul computer usato, al port 901 (se l'installazione l'ha installato lì!): url da introdurre nel browser:

```
http://<Indirizzo o nome del computer>:901/
```

1.15.1 Lato client Windows

Il comando nbtstat (NetBIOS over TCP/IP statistics) dà statistiche sulle connessioni correnti che usano il protocollo NBT.

1.16 HTTP: Apache (port 80)

Apache è il Web server di gran lunga più usato nel S.O. Linux, ed anche quello più usato nel mondo (può girare anche sugli altri Unix e su Windows). E' un server per il protocollo HTTP, per cui il nome del suo demone è httpd.
Apache è un programma modulare, che ha una parte "centrale" relativamente piccola e può caricare a tempo d'esecuzione un gran numero di moduli aggiuntivi, che ne possono espandere grandemente le funzionalità. Per stabilire quali moduli devono essere caricati alla partenza del server e configurare i moduli stessi si possono usare le seguenti direttive, contenute nel file di configurazione del server (httpd.conf):

LoadModule, **AddModule** direttive per indicare quali moduli di estensione compilati separatamente dal server si debbano caricare. L'ordine con cui sono scritti nel file è l'ordine di caricamento ed è importante! Agire con prudenza nel modificare questa parte del file httpd.conf.

Ultimamente Apache è stato "diviso in due"; esistono due eseguibili uno è il server "normale" (httpd) e l'altro è un server specializzato per l'esecuzione degli script Perl (httpd-perl). Dato che ad ogni richiesta da parte di un client viene creato un nuovo processo, normalmente viene lanciato httpd, che è più "leggero" mentre viene lanciato httpd-perl solo quando è richiesto l'interprete perl.

Directory principale

Il directory principale di Apache, consigliato dal team Apache è /usr/local/apache. è peraltro possibile che le distribuzioni lo installino in directory diverse (per sapere dov'è effettivamente provare il seguente comando:

```
# whereis httpd
```

httpd: /usr/sbin/httpd.worker /usr/sbin/httpd /etc/httpd /usr/lib/httpd /usr/share/man/man8/httpd.8.gz

Il risultato visualizzato è quello delle distribuzioni "tipo Red Hat" (Fedora).

httpd.conf

Il file di configurazione di Apache è httpd.conf, che sta in una directory ./httpd/conf del directory principale Apache (vedi ServerRoot). Oltre a questo file potrebbero esserci altri due file di configurazione: access.conf ed srm.conf. L'uso di questi file è però sconsigliato nelle ultime versioni di Apache.
Un altro file di configurazione che è di interesse per Apache è **/etc/mime.types**, che definisce i tipi e sottotipi MIME.

Direttive di httpd.conf

Scrivendo in httpd.conf i nomi di queste direttive, seguiti dai relativi parametri, si configurano tutte le caratteristiche di Apache a tempo di esecuzione. Segue la trattazione delle principali direttive di configurazione di Apache.

ServerType ha parametri: standalone o inetd (xinetd) se si configura con inetd il demone http viene lanciato da inetd (o da xinetd) Altrimenti è un demone "standalone", che deve essere lanciato in modo autonomo e non gode dei servizi di inetd.
Esempio:

```
ServerType standalone
```

Se il server parte con inetd è inetd stesso che deve ascoltare sul port 80 (configurazione in /etc/services), se è standalone ci deve pensare httpd, che ascolta sul port indicato dalla direttiva Port (vedi oltre).
Se si usa inetd per lanciare httpd si può usare il filtro fornito dal TCP wrapper (tcpd). Per configurarlo bisognerà scrivere qualcosa del genere in inetd.conf : `http stream tcp nowait.400 root /usr/sbin/tcpd httpd`

In questo caso però tcpd avvierebbe il demone httpd ad ogni richiesta che giunge dall'esterno, dal lato client, ed il sistema diverrebbe molto più lento. Per questo ogni server che debba funzionare "in produzione" si configura in modo "standalone".

ServerRoot <path di una directory>
è la directory "home di configurazione", dalla quale partono tutte le directory che contengono i file di configurazione, log ed errore di Apache.
Se nel file httpd.conf la path di una di queste directory inizia con un carattere diverso da "/" la directory è relativa a ServerRoot; al nome del file verrà sempre aggiunto "a sinistra" il valore di ServerRoot. Se invece il nome del file inizia con "/" la directory è "assoluta" e verrà usata così come scritta nel file, senza usare ServerRoot.
Esempio:

```
ServerRoot /etc/apache
# SEMPRE SENZA slash alla fine!
```

Da qui in poi le altre direttive fanno riferimento alla path /etc/apache, per esempio scrivere, nel file:

```
ResourceConfig conf/srm.conf
```

equivale a scrivere:

```
ResourceConfig /etc/apache/conf/srm.conf
```

mentre:
```
ResourceConfig /etc/srm.conf
```
indica proprio il file /etc/srm.conf.

ResourceConfig e **AccessConfig**
Per configurare i moduli addizionali caricati con Apache (resource) o per stabilire gli host IP dai quali si accetteranno collegamenti (access) si possono usare due file separati. Se si vuole far così, anche se l'uso è sconsigliato, queste direttive permettono di specificare i file da usare; in questo caso la consuetudine vuole che si usino i file srm.conf e access.conf.
Nelle nuove distribuzioni il file httpd.conf è impostato come se fosse un unico file, ma in realtà ha alcuni "include files" (p.es. in Mandriva c'è l'importante "commonhttpd.conf").

Per indicare che i file access.conf ed srm.conf non sono usati, in httpd.conf possono comparire le righe:

```
ResourceConfig /dev/null
AccessConfig /dev/null
```

L'impostazione Mandriva prevede l'inclusione in httpd.conf del file commonhttpd.conf, dove ci sono tutte le definizioni comuni a tutti i virtual host (cioè a tutti i "siti" indipendenti gestiti dal server). DOPO l'inclusione di commonhttpd.conf in httpd.conf seguono le definizioni dei virtual host, che possono anche scavalcare quelle generali, sostituendole ove necessario.

L'impostazione Red Hat - Fedora non include un file per le definizioni comuni, ma le mette in httpd.conf, verso l'inizio, in una parte detta "Main".

Include <path e nome del file>

"Aggiunge" un altro file di configurazione, che è come se fosse scritto proprio nel punto ove appare questa "Include". Naturalmente se <path e nome del file> comprende uno slash a sinistra esso è "assoluto", altrimenti è relativo a ServerRoot.

1.16.1 Utente che esegue Apache

L'accesso tipico al server Web è "anonimo": ogni accesso di ogni client esterno viene registrato come fatto dall'utente definito dalla direttiva "user" e dal gruppo definito dalla direttiva "group". Per default Apache usa l'utente "noboby", ma spesso le distribuzioni modificano la configurazione "ufficiale" ed usano un utente ed un gruppo "apache". L'utente o il gruppo si possono anche specificare con l'UID (**U**ser**Id**entificator) od il GID (**G**roup **Id**entificator)

Esempi:
```
User apache
Group apache
```
l'utente potrebbe anche essere specificato così:
```
User #100 # è l'utente che ha User Id 100
```

1.16.2 Start e Stop di Apache a "basso livello"

Le distribuzioni mettono a disposizione strumenti visuali per la configurazione di Apache Quando Apache viene lanciato, parte un processo con i diritti di root, il quale, per ragioni di sicurezza ed efficienza, fa partire diversi processi, che girano con i privilegi dell'utente di Apache (utente "apache" nelle distribuzioni RedHat e Mandriva).
Per fermare Apache bisogna fermare il primo processo che è stato lanciato, che fermerà anche tutti i suoi "figli". Questo stesso processo memorizza entro un file il suo Process ID (PID), per cui è facile sapere quale dev'essere il processo da fermare. Per fermare il processo bisogna lanciargli il segnale "TERM" con il comando "kill", per fermarlo e farlo subito ripartire ci vuole il segnale HUP.
Il file nel quale viene memorizzato il PID del processo da "uccidere" è quello specificato nella direttiva PidFile di httpd.conf.
Vediamo dunque un esempio di come si fa:

Riga di httpd.conf (Mandriva 9.X)
```
PidFile /var/run/httpd.pid
```

Per sapere il PID:
```
[root#] cat /var/run/httpd.pid
3476
```

Per fermare Apache:
```
kill -TERM 3476
```

Per fermare Apache e farlo ripartire:
```
kill -HUP 3476
```

Apache crea un processo per ciascuna richiesta di connessione da parte dei client. Inoltre tiene aperti un certo numero di processi "di riserva" cui assegnare rapidamente le nuove connessioni che arriveranno. Per ottimizzare le prestazioni del server è possibile stabilire il numero minimo e massimo di questi processi.

MinSpareServers e **MaxSpareServers** (spare = "di riserva") sono il numero minimo e massimo di processi che rimangono aperti in memoria in attesa di essere usati come server. **StartServers** è il numero iniziale di questi processi.

MaxClients è il numero massimo di server che possono girare contemporaneamente.

Keep_Alive; Keep_Alive è un'estensione al protocollo HTTP, presente a partire da HTTP 1.1, che permette connessioni "persistenti". Se Keep_Alive è abilitato il server non chiude i collegamenti TCP alla fine della trasmissione degli oggetti ipertestuali (file html, immagini), ma li mantiene aperti e questo può implicare, se le pagine hanno molte piccole immagini, un notevole incremento delle prestazioni del server. I parametri di Keep_Alive, che si possono configurare sono: KeepAlive, KeepAliveTimeout, MaxKeepAliveRequests

Listen [<indirizzo IP>:]<numero di port> permette di specificare un numero di port od anche una coppia indirizzo:port sul quale il demone rimane in attesa di richieste di collegamento TCP. In httpd.conf si possono specificare più di una riga "Listen". Naturalmente è logico specificare 80, il port di default per http definito dagli standard Internet. Se non è presente una direttiva Listen il server parte, ma non risponde mai a nessuna richiesta di collegamento.

Esempio:
```
Listen 80   # port standard HTTP (well known port number)
Listen 8080 # port tipico del proxy http
```

Le due direttive precedenti fanno ascoltare il server su tutte le sue schede di rete ed indifferentemente sui port 80 ed 8080.
Con Listen si può fare in modo che lo stesso server Apache ascolti su più di un port TCP, "personalizzando" il port in base all'indirizzo IP della scheda di rete da cui ha raccolto la richiesta.

Esempio:

```
# per l'indirizzo del server sulla rete 192.168.13.0 uso il port standard
Listen 192.168.13.25:80
# per l'indirizzo del server sulla rete 172.16.0.0 uso il port "da proxi"
Listen 172.16.13.8:8080
```

Si noti che gli indirizzi di Listen devono essere indirizzi IP validi di una delle schede di rete presenti nel computer che ospita Apache.

1.16.3 Risorse

Un tempo la definizione delle risorse era fatta all'interno del file srm.conf, oggi si consiglia di metterla in httpd.conf od in un file include.
Comandi che configurano le risorse: DocumentRoot, DirectoryIndex, DocumentDir, UserDir, IndexIgnore, Alias, ScriptAlias, Access <filename>

DocumentRoot <directory> [<nomedi file>] è la directory dalla quale parte il contenuto del sito (es. le pagine HTML). Ogni volta che Apache si procura la path di un documento che fa parte del sito Web aggiunge DocumentRoot a sinistra della path data nell'URL.
Per esempio se l'URL è:

```
http://ingmonti.it/libri/index.php
```

e in httpd.conf (o in commonhttpd.conf) compare una linea come questa:
```
DocumentRoot "/var/www/html"
```
il server fornirà al browser il file /var/www/html/libri/index.php

BindAddress <indirizzo IP o nome host> | * se il server ha più di un indirizzo IP (per esempio se ha più di una scheda di rete od ha degli alias su una scheda) limita il suo funzionamento solo sull'indirizzo IP indicato. Con questa direttiva si può fare in modo di avere più server Apache sulla stessa macchina, ciascuno con un BindAddress diverso.

ServerAdmin <indirizzo di posta elettronica> l'indirizzo e-mail dell'amministratore del server Apache. L'indirizzo specificato verrà usato nelle pagine d'errore che il server genera automaticamente, in modo che gli utenti possano notificare eventuali errori al Webmaster. L'indirizzo potrebbe anche essere usato per mandare automaticamente messaggi di posta elettronica in occasione di errori nel server.

ServerName <nome di dominio> | <indirizzo IP> è il nome "ufficiale" del server, con cui il server si identifica. Esso viene spedito al client quando il server risponde. Deve essere un nome DNS valido. Può essere un nome diverso rispetto all'hostname del computer su cui Apache sta girando. Se non viene specificato nulla Apache usa il nome del computer.

Errorlog e **Transferlog** direttive che specificano i directory in cui debbono essere memorizzati i file di log degli errori e dei collegamenti.

DirectoryIndex <lista di file>
Questa direttiva permette di configurare quali file il server cerca automaticamente se viene indicato nella URL un nome di directory "da solo" senza nessun nome specifico di file. DirectoryIndex fa cercare i file della lista indicata quando viene specificata solo la directory ed il nome del file manca. Un file "Directory Index" è dunque un file che viene fornito all'utente se esso specifica solo un percorso, senza indicare un file preciso.

Esempio:
```
DirectoryIndex index.html index.shtml default.htm index.php index.cgi
```

Con questa direttiva all'interno di un blocco, Apache cerca nella directory i file indicati nell'ordine indicato. Il primo che viene trovato viene spedito all'host che ne ha fatto richiesta. Se Apache non trova nessuno dei tre file indicati, dà:

- un file HTML generato dinamicamente che fa vedere tutti i file contenuti del directory specificato dall'URL

oppure

- un file HTML che dà al richiedente un'indicazione di errore, quale, per esempio: "Not Found; the requested URL was not found on this server". La pagina fornita si può cambiare configurando Apache.

Per decidere il comportamento di Apache quando non esiste un file Directory Index si usa la direttiva IndexIgnore:

IndexIgnore <schema per la selezione di file> esclude dalla visualizzazione del contenuto dei directory i file che rispettano lo <schema per la selezione di file> (pattern). Il comportamento del server riguardo alla visualizzazione dei directory può essere modificato anche "directory per directory", con l'Opzione "Indexed" della Sezione <Directory> di httpd.conf (vedi oltre).

Redirect <path dell'URL vecchio> <nuovo URL>
Quando viene richiesto dal client un URL, all'interno di "questo" server, che fa parte del <path dell'URL vecchio> il server ridirige la richiesta al <nuovo URL>, che può anche essere all'esterno di questo server, chiedendo un file dello stesso nome di quello originariamente richiesto dal client.
è utile per cambiare di "posto" ad un sito senza perdere i link ad esso.
<nuovo URL> deve essere completamente specificato (http://..) tranne naturalmente nel nome del file, che manca, invece <path dell'URL> è solo la parte di path che parte da DocumentRoot, dato che fa riferimento a "questo" Web server.
Esempio:

```
Redirect /dati2003 http://netBackup.com/acme-srl/2003
```

se il client chiede http://nomeDelMioServer/dati2003/vendite.html
il server fa puntare il browser del client a http://netBackup.com/acme-srl/2003/vendite.html

Alias <directory dichiarata nell'URL> <directory usata da Apache>
permette di usare sull'hard disk directory diverse da quelle richieste dal browser per mezzo dell'URL. Nello stesso file httpd.conf è possibile usare molte direttive Alias.

Esempi:

```
Alias /2003 /computerBackup/dati2003
```

Se il client chiede http://nomeDelMioServer/2003/vendite.html il server dà il file /computerBackup/dati2003/vendite.html
Se computerBackup è uno share di un server di rete montato in quel directory, i dati che verranno forniti al client vengono presi attraverso la rete locale.

ScriptAlias < .. > < .. >
Fa lo stesso di Alias per quelle directory che contengono script.

TypesConfig <path e file> dice in quale file si trovano le definizioni dei tipi MIME.

HostnameLookups se ON fa entrare i client solo se è possibile fare un "reverse lookup" nel DNS dell'indirizzo IP da cui proviene la richiesta.

1.16.4 Accesso

La definizione delle modalità di accesso ai directory, che una volta veniva scritta nel file access.conf, permette di realizzare un server "non anonimo", proteggendo il server e controllando che l'accesso ai file sia permesso a solo chi ne ha il diritto. Le direttive di accesso sono divise in "sezioni", ciascuna delle quali inizia e finisce fra tag in "stile HTML".

Le sezioni sono: **<Directory>** e **<File>** (che controllano l'accesso agli interi directory od ai singoli file), **<Location>**, che configura l'accesso agli URL passati dal client, **<Limit>** che può limitare a particolari utenti l'utilizzazione di specifici metodi HTTP (GET, HEAD, POST, PUT, DELETE).

Sezioni Directory e File, sintassi generale:

<Directory <espressione regolare>**>**

<specifiche che si applicano alle directory che "matchano" l'espressione regolare>

</Directory>

<File <espressione regolare> **>**

<specifiche che si applicano ai file che "matchano" l'espressione regolare>

</File>

L'espressione regolare determina una o più directory (o file) che verranno trattate come indicato dalle specifiche che seguono nel blocco (<specifiche>).
Le specifiche si applicano anche ai sottodirectory di quelli definiti dall'espressione regolare. Le direttive più interessanti che si possono usare dentro ai blocchi Directory o File sono: Options, Order, Allow from, Deny from, AllowOverride
Direttive Allow e Deny, sintassi:

Allow from <lista degli host concessi>

e

Deny from <lista degli host vietati>

la lista di host è composta dalla parola "all", oppure da un elenco di host, indicati con l'indirizzo IP, eventualmente "semplificato", oppure come nome DNS, ma in questo caso solo se Hostnamelookup viene messo ON. Gli indirizzi semplificati si indicano senza includere la parte di indirizzo che indica gli host che si vogliono "raggruppare" (es. 197.39.), oppure scrivendo, dopo l'indirizzo IP e dopo una barra (slash) una maschera di sottorete.

è necessario configurare l'ordine con il quale si vuole che vengano lette le direttive allow e deny, con la direttiva **Order**

Esempio:
```
Order allow, deny
```
esegue prima la direttiva allow (permetti), poi la deny (nega) Direttiva Option, sintassi:

Option [+|-] <opzione> [[+|-] <opzione>] ..

Le opzioni sono specificate con un nome e sono precedute da un + (opzionale) per abilitarle o da un - per disabilitarle.
Le opzioni specificano il comportamento del server riguardo alla sezione cui si applicano.
Alcune delle opzioni sono:
All attiva tutte le opzioni, tranne MultiView
Indexes quando l'opzione è attivata (+), se il server non trova nel directory relativo alla sezione nessun file fra quelli indicati in DirectoryIndex, fornisce al browser un file HTML che fa vedere il contenuto del directory. L'utente
FollowSymLinks "segue" i link simbolici; il server interpreta i link ad altri file (symlink, collegamenti) come se fossero veri file. L'uso di link simbolici all'interno delle directory Web è da sconsigliare, perché abbassano il livello di sicurezza del sistema quando vengono "seguiti". Per questo la configurazione di default dei directory è meglio che preveda che i link simbolici non vengano seguiti, a meno di non prevedere anche altre restrizioni che limitino l'accesso a persone "fidate".
ExecCGI quando l'opzione è attivata i file eseguibili contenuti in questo directory possono essere eseguiti.
Esempi:
```
<Directory />
    # "/" ^ qui sopra è la directory root, essa deve essere fortemente protetta,
    # impedendone l'accesso a chiunque.
```

```
    # Altre sezioni Directory, successive nel file httpd.conf, "apriranno"
    # l'accesso agli specifici directory che si vorranno usare per i nostri siti
    #
    # Questa è la configurazione del directory root consigliata dal team Apache,
    # che blocca proprio tutto:
    Options -All -Multiviews
    AllowOverride None
    Order deny, allow
    Deny from all
</Directory>

<Directory /var/www/cgi-bin>
    # questa è la directory tipica ove si mettono i programmi lato server,
    # per cui Apache dovrà consentire l'esecuzione di programmi
    # da questa directory
    AllowOverride All
    Options ExecCGI
</Directory>

<Directory /var/www/scriptDelCapo>
    # questa è una directory che può contenere programmi,
    # ma che può essere usata solo da una
    # particolare stazione, appartenente alla rete locale
    AllowOverride All
    Options ExecCGI
    Order deny,allow
    Deny from all
    Allow from 192.168.26.133
</Directory>

<Directory /var/www/scriptDelGruppo>
    # questa è una directory che può contenere programmi,
    # ed il cui directory può essere mostrato via Web.
    # Essa può essere usata solo da una particolare rete locale
    # (la rete 192.168.26.0)
    # dall'indirizzo 172.16.12.48
    # e da un indirizzo IP che proviene dal dominio DNS ingmonti.it
    # (verrà fatto un reverse lookup DNS)
    AllowOverride All
    Options Indexes ExecCGI
    Order deny, allow
    Deny from all
    Allow from 192.168.26. 172.16.12.48
    Allow from .ingmonti.it
</Directory>

<Directory "/var/www/html/prova">
 # permette l'accesso a tutti gli host della rete di classe B
 # 172.16.0.0 (la maschera usata è "da classe B") ed a tutti gli host
 # della rete di classe A 10.0.0.0 che hanno i primi tre Byte
 # uguali a 10, 11 e 12 (10.11.12.0, la maschera NON è quella
 # standard di classe A)
 Order deny,allow
 Deny from all
 Allow from 172.16.0.0/255.255.0.0 10.11.12.0/255.255.255.0
</Directory>

<Directory /var/www/soloCertiIP>
    # con il mascheramento, questa directory verrà riservata
    # solo agli host con indirizzo IP fra 192.168.13.128 e 192.168.13.159
    Order deny, allow
    Deny from all
    Allow from 192.168.13.128/255.255.255.224
    # Infatti la maschera 255.255.255.224 corrisponde, nell'ultimo byte,
    # a 11100000 in binario, e metterebbe nella stessa rete
    # (e quindi darebbe accesso al directory) tutti gli host con indirizzo
    # YYYXXXXX. L'indirizzo 192.168.13.128 stabilisce che YYY è 100
    # (infatti 128 è 10000000), per cui gli indirizzi accettati hanno,
```

```
    # nell'ultimo Byte, da 10000000 a 10011111 in binario, cioè vanno
    # da 192.168.13.128 a 192.168.13.159
</Directory>

<Files ~ "^\.ht">
    # questa sezione si applica a tutti i file il cui nome
    # segue la regular expression indicata,
    # cioè che iniziano con le lettere .ht
    Order allow,deny
    Deny from all
    # questa serve per non dare a nessuno i file tipo .htaccess,
    # che contengono i diritti di accesso degli utenti ai file
    # di ogni directory.
</Files>

<Files "*index*">
    # questa sezione si applica a tutti i file il cui nome contiene "index"
    # e non fornisce questi file solo se la richiesta proviene dal computer
    # che ospita il server
    Order deny,allow
    Deny from 127.0.0.1
    # (questa sezione può servire in sede di diagnostica)
</Files>
```

1.17 Virtual Host

Apache permette di gestire siti completamente indipendenti sullo stesso server. Gli host virtuali possono essere basati sull'indirizzo IP o sul nome. Nel primo caso ogni virtual host ha indirizzo IP diverso ed un solo virtual host può stare su quell'indirizzo.

Gli host basati sull'indirizzo IP distinguono fra gli host virtuali per mezzo dell'indirizzo IP; quelli basati sul nome usano invece informazioni supplementari passate dal client HTTP. Il browser infatti può includere il nome DNS dell'host fa la richiesta nell'header HTTP[1]. In questo modo si possono avere diversi host virtuali sullo stesso indirizzo IP. Gli host basati sul nome non possono essere usati con il protocollo sicuro SSL.

I Virtual Host di Apache possono essere configurati per girare su processi diversi o su un singolo demone.

- più demoni http (httpd)
  i server http sono completamente separati, con file di configurazione e directory dati completamente diverse
  sono richieste molte risorse al sistema
- unico demone http
  configurazione più "leggera"; qui si considererà solo questo caso

Per configurare un virtual host si definisce una sezione <VirtualHost>. Per configurare un Virtual Host basato sul nome è necessario configurare un server DNS locale, che fornirà ad Apache l'indirizzo IP del nome di domino richiesto dal browser.

All'interno di una sezione <VirtualHost> si possono definire le direttive per separare il sito dagli altri. Si possono usare per esempio le direttive: ServerAdmin per stabilire l'indirizzo e-mail del Webmaster, ServerName per il nome con cui il server si identifica, DocumentRoot per avere una home page in un posto diverso dagli altri siti, ErrorLog per memorizzare gli errori di accesso a questo sito in un posto diverso dagli altri. All'interno della sezione <VirtualHost>

si possono usare quasi tutte le direttive di Apache; ciò rende la configurazione molto flessibile.

Esempio di virtual host basato sull'indirizzo IP

```
<VirtualHost 10.11.12.13>
    DocumentRoot /www/serverA
</VirtualHost>
```

Esempio di server che risponde a due indirizzi con lo stesso contenuto:

```
<VirtualHost 192.168.13.8 172.16.12.48>
    DocumentRoot /www/comune
</VirtualHost>
```

Esempio di server che dà due siti diversi in base al port usato

1 Questa caratteristica dell'header HTTP è obbligatoria nello standard a partire dalla versione HTTP 1.1.

```
Listen 80
Listen 8080
<VirtualHost 192.168.13.8:80>
    DocumentRoot /www/sitoPort80
</VirtualHost>

<VirtualHost 192.168.13.8:8080>
    DocumentRoot /www/sitoPort8080
</VirtualHost>
```

Esempio di host virtuale basato sul nome:

Per usare un virtual host basato sul nome è necessario specificare su quali indirizzi IP Apache dovrà attendersi richieste basate sul nome; ciò viene fatto per mezzo della direttiva NameVirtualHost.
Se nella sezione <VirtualHost> è presente la direttiva ServerName l'host virtuale è basato sul nome.

```
NameVirtualHost 172.16.13.8
# direttiva che stabilisce gli indirizzi IP utilizzati dal server Apache
# per tutti gli host virtuali basati sul nome
#  (mettere * al posto dell'indirizzo IP per usare tutti
#   gli indirizzi IP del computer su cui gira il server)
#   Solo gli indirizzi indicati qui saranno usati dagli host virtuali

# segue la definizione di un singolo host virtuale:
<VirtualHost 172.16.13.8>
        # la specificazione dell'indirizzo, data qui sopra, deve essere
        # identica a quella data nella direttiva NameVirtualHost

        # mettendo una direttiva ServerName in questa sezione,
        # il ServerName per questo host virtuale è diverso da quello
        # generale di Apache. Questa direttiva definisce che questo
        # virtual host è basato sul nome
        ServerName www.ingmonti.it

        # si possono definire dei nomi "di alias":
        ServerAlias ingmonti.it libri.ingmonti.it
        # il server erogherà al client le stesse pagine,
        # se richieste con uno dei
        # seguenti nomi di dominio:
        # 1. www.ingmonti.it
        # 2. ingmonti.it
        # 3. libri.ingmonti.it
        # naturalmente il server DNS deve esser configurato
        # in modo che indirizzi verso un indirizzo IP del nostro
        # server tutte le richieste che riguardano i nomi 1.,2. e 3.

        # questo server virtuale può avere la sua path DocumentRoot,
        # diversa da quella "generale":
        DocumentRoot /www/sito2
</VirtualHost>
```

Altri host possono essere configurati analogamente.

Per verificare la sintassi delle direttive di definizione dei virtual host si può lanciare httpd con l'opzione -S.

è possibile fare in modo che il server risponda autonomamente con pagine in lingue diverse, in base alla lingua dichiarata dal browser (vedi il manuale di Apache, che viene installato insieme al package).

1.18 Autenticazione, autorizzazione, controllo d'accesso

L'autorizzazione all'accesso in certe directory può essere condizionata alla digitazione della password corretta (autenticazione dell'utente).
Quando una directory è "protetta" il server chiede al browser che tenta di accedervi l'"autenticazione", che avviene tramite la richiesta di una password all'utente da parte del browser. La password digitata viene poi passata dal browser al server Apache, che la controlla e garantisce l'accesso se essa è valida. Dato che HTTP è "senza stato", ciò accade per OGNI acceso ad OGNI risorsa protetta, anche se nella maggior parte dei casi l'utente non se ne accorge, perché a passare la password al server ogni volta ci pensa il browser "di nascosto" e la chiede all'utente solo la prima volta.

Perché l'autenticazione funzioni è necessario che sia caricato il modulo Apache "auth_module", per questo devono essere presenti nel file httpd.conf i comandi:

LoadModule auth_module modules/mod_auth.so
e
AddModule mod_auth.c

Apache ha una SUA gestione degli utenti, completamente diversa da quella di Unix o Samba.
Gli utenti e le password Apache si creano usando il programma htpasswd.
Sintassi:

htpasswd <opzioni> <file delle password> <username>

L'opzione più importante è:
-c = create; crea da zero o sovrascrive una tabella delle password (se esiste già la distrugge!)
Per esempio, per creare "da zero" il nuovo file delle password "/etc/segretissimo" contenente l'utente "capo" si può fare così:

```
[root#] htpasswd -c  /etc/segretissimo capo
```

htpasswd chiede la password all'utente, e la memorizza criptata nel file /etc/segretissimo

Per aggiungere nuovi utenti si fa lo stesso, ma non si deve mettere l'opzione -c. Esempio:

```
[root#] htpasswd /etc/segretissimo fantozzi
```

Per ragioni di sicurezza bisognerebbe limitare la proprietà ed i diritti di lettura del file delle password soltanto all'utente con cui gira apache. Esempio:

```
[root#] chown apache.apache /etc/segretissimo

[root#] chmod 640 /etc/segretissimo
```

Una volta creato il file delle password si deve configurare Apache per usarlo, e ciò si può fare in due modi:
1. scrivendo nel file httpd.conf una sezione <Directory>, relativa al directory che si vuole proteggere, che contenga le direttive di autenticazione, che sono AuthType, AuthName, AuthUserFile, AuthGroupFile e Require.
2. piazzando un file .htaccess dentro ciascuno dei directory che si vuole difendere con l'autenticazione. Il file .htaccess conterrà le direttive di autenticazione, in modo analogo ad una sezione Directory

1.18.1 Direttive di autenticazione

AuthUserFile identifica il nome del file delle password degli utenti

Il file con l'elenco degli account creati con htpasswd deve essere indicato questa direttiva dentro questa sezione (AuthUserFile). Nell'esempio precedente:

```
AuthUserFile /etc/segretissimo
```

AuthName identifica con un nome l'area di autenticazione ("realm" o "reame" di autenticazione"). Questa stringa verrà comunicata al browser al momento della richiesta di autenticazione, ed esso lo mostrerà come "titolo" della finestra di login.

AuthTypeBase identifica il tipo di autenticazione; il valore più tipico in questo caso è Base, altre opzioni sono complicate.

AuthGroupFile indica il file di testo che contiene i gruppi di utenti (groupfile) .Un groupfile è un semplice file di testo che contiene righe che indicano il nome dei gruppi e ciascuno dei loro componenti.
Una riga ha la forma:
<Nome gruppo>**:** <lista di nomi utente>

Per esempio, un contenuto di file di gruppo potrebbe essere il seguente:

```
contratti: bianchi rossi verdi
ufficiotecnico: mari monti collina
```

Require <elenco degli utenti che hanno diritto ad accedere>
Elenco di username di utenti, presenti nel file "AuthUserFile", che possono accedere. Gli username dell'elenco sono se-

parati da spazi.
Se tutti gli utenti vanno bene si specificherà:

```
Require valid-user
```

Se si vorrà riservare l'accesso ai soli membri di un gruppo:

```
Require group <nome del gruppo all'interno del file definito con AuthGroupFile>
```

Esempio di una sezione Directory che richiede l'autenticazione:

```
<Directory /home/httpd/html/dirigenti>
        Option none
        AllowOverride none
        Order deny, allow
        Allow from all
        AuthName "Accesso riservato all'area contratti"
        # viene spedito al client che così può sapere automaticamente
        # qual è il nome d'utente da usare per accedere a questo server
        AuthType Basic
        AuthUserFile /etc/segretissimo
        AuthGroupFile gruppi
        Require group contratti
        # possono entrare solo gli utenti del gruppo contratti
</Directory>
```

Naturalmente il file "gruppi" sarà cercato sotto ServerRoot (non ha "/" a sinistra). Si darà accesso solo agli utenti che sono nella lista degli utenti del gruppo "contratti".

Aggiungere i directory degli utenti al Web gestito da Apache

Se in httpd.conf si scrive

UserDir <nome di directory del server da rendere "pubblico">

gli utenti potranno creare, nei loro home directory, una directory con quel nome e riempirla con pagine HTML ed altri contenuti Web. Apache renderà visibile dal Web quella directory, che sarà la "root" dei Web "di utente". Per accedere a quel directory bisognerà usare l' URL:

di dominio Internet del server>/~<username dell'utente>

Si può configurare un modo diverso di indicare la directory d'utente.

Esempio:

```
UserDir public_html
```

Se l'utente gamon crea la directory /home/gamon/public_html essa sarà accessibile da Web all'URL:

http://dominiodelserver.it/~gamon

Si dovranno assegnare i diritti sulle directory public_html in modo che esse siano accessibili all'utente specificato come "User" in httpd.conf (tipicamente nobody o apache).

è possibile indicare una lista di nomi d'utente cui è impedita la costruzione di siti Web nei loro home directory. Ciò si ottiene con la direttiva DISABLED.

Esempio:

```
user DISABLED gamon gmonti ufantozzi
```

Pagina di benvenuto

Se non è definita nessuna home page (non c'è nessun file "Index" nella DocumentRoot) Apache produce la "Welcome page", che in Red Hat si può configurare con il file di include Welcome.conf in /etc/httpd/conf.d.

Apache manual

Apache comprende un manuale, in HTML, che di solito viene configurato per rispondere alla directory virtuale <Nome del server>/manual/.

CGI

HTPP permette, oltre allo scambio di documenti ipertestuali, anche lo scambio di dati, attraverso un meccanismo legato all'esecuzione di particolari **programmi**, detti "**gateway**".
Questi programmi possono essere scritti in qualsiasi linguaggio, basta che rispettino un'interfaccia di programmazione standard detta "**C**ommon **G**ateway **I**nterface" (CGI).
Il modo più diretto di rispettare la CGI è scrivere semplici shell script che ricevono input attraverso lo standard input o particolari variabili d'ambiente e spediscono al client pagine formattate in HTML, attraverso lo standard output (p.es. con comandi "echo").

Il server HTTP riceve un URL. Se quell'URL "punta" al directory definito come ScriptAlias in httpd.conf il server sa che non è un file qualsiasi, ma un comando da eseguire, ovverosia un programma eseguibile per uno script interpretato). Perciò il server esegue il comando, passandogli l'URL come parametro. Attraverso l'URL, oppure attraverso il file di standard input, il programma CGI è in grado di ricevere dei dati in ingresso, che provengono dal client (browser).
A questo punto il programma CGI può eseguire le sue elaborazioni, per poi restituire al client, attraverso il suo standard output, un file HTML.
Il browser interpreta quel file come se fosse una normale pagina HTML "statica", senza neppure rendersi conto che in verità è il risultato dell'esecuzione di un programma.

Il server Apache può riconoscere i file da eseguire non solo perché stanno nel directory ScriptAlias, ma anche quando il loro nome porta una particolare estensione, definita con la direttiva AddHandler in httpd.conf. Va notato che questo approccio potrebbe rivelarsi poco sicuro, perché QUALSIASI file che termina con estensioni dichiarate eseguibili verrà eseguito, anche programmi che vengono lanciati "con malizia" da utenti che vogliono prendere il controllo del computer.

La risposta dei programmi CGI deve rispettare in tutto e per tutto i dettami imposti da HTTP; deve perciò iniziare con la descrizione del contenuto MIME, con la riga "content_type", poi deve seguire una riga vuota e successivamente il contenuto del file, che potrebbe essere un file HTML, ma anche un file di tipo binario come, per esempio, una immagine od un suono.

Uno script di esempio:

```
#!/bin/sh # specifica che si usa l'interprete di comandi sh (shell)
echo "content_type: text/html"
echo # linea vuota!
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE> Script CGI di prova </TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1> Script CGI di prova </H1>"
whoami # comando di sistema che scrive nello standard output
echo "</BODY>"
echo "</HTML>"
```

Questo testo va salvato nel directory definito come "ScriptAlias" nel file di configurazione Apache e reso eseguibile. Supposto che il file si chiami "primoCGI.sh", esso verrà lanciato quando il browser richiederà l'URL:

<host>/<Directory ScriptAlias>/primoCGI.sh

Configurare Apache per i CGI

Apache ha un modulo per il CGI, che deve essere abilitato con:

LoadModule cgi_module /modules/mod_cgi.so
e
AddModule mod_cgi.c

Variabili d'ambiente del server

Le variabili d'ambiente del server si usano come tutte le normali variabili degli script e specificano le caratteristiche della connessione che si instaura fra client e server. Alcune variabili d'ambiente possono contenere stringhe che il client passa al server come parametri della URL.

```
SERVER_SOFTWARE
SERVER_NAME
GATEWAY_INTERFACE
SERVER_PROTOCOL
SERVER_PORT
REQUEST_METHOD
HTTP_ACCEPT
```

```
PATH_INFO = "$$$$$$"
PATH_TRANSLATED = "$$$$$$"
SCRIPT_NAME = "$$$$$$"
QUERY_STRING = "$$$$$$"
REMOTE_HOST = "$$$$$$"
REMOTE_ADDR = "$$$$$$"
REMOTE_USER = "$$$$$$"
echo AUTH_TYPE = "$$$$$$"
echo CONTENT_TYPE = "$$$$$$"
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

1.18.2 Passaggio di parametri ai programmi CGI

IsIndex e **IsMap**

Sono due tecniche per chiedere un input all'utente attraverso il browser. IsIndex serve per gli input di testo ed oggi non è più usato, dato che si usano i FORM, IsMap dà la posizione ove l'utente ha fatto click con il mouse entro un'area nella quale viene visualizzata un'immagine. Lo script deve ricavare dalla variabile QUERY_STRING l'informazione spedita tramite IsIndex o IsMap.

Moduli FORM

Il browser genera coppie <nome del campo> = <valore del campo> e spedisce il modulo di testo corrispondente al programma CGI.
Il browser genera un modulo FORM quando incontra il tag HTML <FORM>.

Sintassi

<FORM <parametri di FORM>**>**

<descrizione del form>

</FORM >

Principali <parametri di FORM>

ACTION permette di individuare l'URL cui il form viene spedito (deve comprendere il nome dello script CGI)

METHOD specifica il modo della spedizione, che può essere GET o POST

Esempio di HTML per GET:

```
<FORM ACTION="/cgi-bin/input_superiori.sh" METHOD="GET">
```

Esempio di HTML per POST:

```
<FORM ACTION="/cgi-bin/input_inferiori.sh" METHOD="POST">
```

GET e **POST**

Con GET le informazioni sono aggiunte alla fine dell'URL, separandole con un punto interrogativo. Se le informazioni passate contengono un "=" provengono da un FORM.

Esempio di URL per GET, che spedisce i dati di un form:

http://megaditta/cgi_bin/dati_inferiori.sh?nome=Ugo&cognome=FANTOZZI

da questo esempio si può vedere come le coppie <nome del campo>=<valore del campo> siano separate dal carattere "&".

L'utilizzazione del metodo GET ha il vantaggio principale che i dati spediti sono "memorizzabili" attraverso un segnalibro del browser e perciò gli utenti potranno ripetere la stessa query in un secondo tempo in modo molto facilitato.

Peraltro se i dati sono molti, è preferibile usare il metodo POST, che non utilizza la URL per il passaggio dei dati. Infatti con questo metodo i dati sono spediti allo script CGI attraverso il normale stream "file" di standard Input del programma CGI. I dati sono spediti nello standard input formattati come se provenissero dall'URL, cioè in modo identico ai parametri che invece arrivano attraverso la URL.

Naturalmente, a differenza di un GET, un metodo POST è "privo di stato" e non si può salvare in un bookmark.

Con il metodo POST il programma CGI deve prendere i dati che provengono dallo standard input ed accedere alle singole informazioni interpretando ciò che arriva. Il linguaggio PERL è molto usato per questo scopo, perché è molto versato al "parsing" delle stringhe. Per i programmi CGI possono essere usati anche molti altri linguaggi o strumenti software, quali PHP, Javascript, awk, VBscript (solo in S.O. Microsoft).

Esempio di form con "text box" e metodo GET

```
<HTML>
<HEAD>
<TITLE> File HTML con un FORM contenente text box </TITLE>
</HEAD>
<BODY>
<FORM ACTION="/cgi-bin/input_superiori.sh" METHOD="GET">
Nome <INPUT TYPE="text" NAME="Nome" VALUE="Umberto" > <br>
Cognome <INPUT TYPE="text" NAME="Cognome" VALUE="MAZZANTI VIENDALMARE" > <br>
Titolo <INPUT TYPE="text" NAME="Titolo" VALUE="Conte Duca"> <br>
<INPUT TYPE="submit" value="Spedisci i dati">
</FORM>
<BODY>
</HTML>
```

INPUT TYPE=submit significa che i dati vengono spediti quando l'utente preme il tasto visualizzato dal browser. In questo caso il tasto contiene la scritta "spedisci i dati".

Se il form precedente viene spedito senza modificare i dati di default, con un semplice click sul bottone, ciò che giunge al programma CGI /cgi-bin/input_superiori.sh, attraverso il port HTTP riservato alla comunicazione delle URL, è:

```
http://megaditta/cgi_bin/input_superiori.sh?Nome=Vittorio+Emanuele&Cognome=MAZ-
ZANTI+VIENDALMARE&Titolo=Conte+Duca
```

Se nelle stringhe ci sono dei caratteri speciali essi vengono codificati con il loro codice ASCII in esadecimale, preceduto da "%".

Esempio di form con "combo box" e metodo POST

```
<FORM ACTION="/cgi-bin/input_inferiori.sh" METHOD="POST">
Scegliere il tipo di punizione:
<SELECT NAME="selezione">
<OPTION SELECTED VALUE=1> In ginocchio sui ceci
</OPTION>
<OPTION VALUE=2> Cancellazione ferie </OPTION>
<OPTION VALUE=3> Crocifisso in sala mensa </OPTION>
</SELECT>
<br>
<INPUT TYPE="submit" value="Spedisci i dati">
</FORM>
```

Seguono due esempi molto semplici e del tutto inutili che dovrebbero spiegare chiaramente l'Input Output dei programma CGI.

Esempio di programma CGI per metodo GET

Supponiamo che il seguente sia il codice del programma input_superiori.sh, invocato dal precedente form HTML, che faceva uso del metodo GET.

```
#!/bin/sh
# script CGI di prova
echo "Content-type: text/html"
echo
# linea vuota!
# fine header HTTP

# visualizza la QUERY_STRING così come viene ricevuta:
echo "Eccellentissimo $QUERY_STRING
"

echo "Siamo felici di comunicarCi che il suo premio produzione è di Euro 1 000
000"
```

Il risultato dell'esecuzione di questo programma è spedito al browser con le echo, attraverso lo standard output del programma CGI. Ciò che si vede sul browser è il seguente:

```
Eccellentissimo Nome=Vittorio+Emanuele&Cognome=MAZZANTI+VIENDALMARE&Titolo=Con-
te+Duca
```

```
Siamo felici di comunicarCi che il suo premio produzione è di Euro 1 000 000
```

Questo esempio mostra come il programma CGI ottenga i dati spediti con il metodo GET nella sua variabile d'ambiente QUERY_STRING. Sarà suo compito, e qui non lo vedremo, interpretare quella stringa in modo corretto.

Esempio di programma CGI per metodo POST
Il seguente sia il codice del programma "input_inferiori.sh", usato per comunicare con il form HTML che faceva uso del metodo POST.

```
#!/bin/sh
# script CGI di prova
echo "Content-type: text/html"
echo                    # questa è una linea vuota!
# fine header HTTP

# lo script più semplice per il metodo POST.
# visualizza il file di standard input sul file di standard output:
cat
```

l'unica istruzione che interviene sullo standard output è cat, che, usata senza alcun parametro, prende il suo ingresso dallo standard input (normalmente lo prende da un file, ma se il file non c'è ..).
dopo aver preso lo standard input cat lo trasmette, come sempre, nello standard output, per cui il risultato del programma CGI è di rimandare indietro al browser ciò che esso ha spedito al server.

Questi due semplici esempi illustrano quanto la vita sia difficile per i programmi CGI. Con i programmi CGI non si conosce a priori né quanti sono i campi presenti in un form, né il loro nome, oltre che, naturalmente, il loro contenuto. Inoltre i dati arrivano tutti "insieme" e bisogna darsi da fare per separarli. La programmazione è dunque complicata, anche se scrivendo qualche funzione la si rende molto più semplice.
Per ovviare a questi problemi vengono in aiuto strumenti che fanno facilmente il parsing[1] delle stringhe provenienti dal browser. Storicamente il primo linguaggio utilizzato in questo campo, nato proprio per questo tipo di lavoro, è il PERL, ma ultimamente si preferisce usare strumenti più facili, quali PHP (su tutti i server), C# o VBscript (solo sui server Microsoft).

1.18.3 Proxi Apache

Il server Apache può caricare il modulo "proxi_module" che svolge alcune semplici funzioni da proxi. Se però si vuole un "vero" e completo programma per un proxi HTTP, allora si può usare il programma "squid".

2 Complementi

2.0.1 inetd e xinetd

inetd (internet daemon) è il demone che fa partire e gestisce tutti i servizi TCP/IP, cioè i "server" ed i "client" di tutti i protocolli internet. In passato era l'unico punto di partenza di tutti i protocolli di livello applicazione (server). All'ingresso in un nuovo runlevel, partiva inetd (o la sua versione più recente xinetd) e lanciava tutti i server internet. In questo modo era possibile una gestione più "centralizzata" del protocollo TCP/IP, che fa da base ai server.
Recentemente l'approccio è cambiato ed i server più importanti tendono a partire "da soli", in modo completamente indipendente da inetd. inetd e xinetd gestiscono ancora protocolli meno usati, fra i quali il più importante è telnet.
inetd è un processo di sistema che risponde ad ogni richiesta di connessione TCP, per qualsiasi protocollo. Se la richiesta deve essere accettata questo il processo inetd crea un altro processo, specifico per il protocollo richiesto, che gestisce il collegamento TCP relativo a quel "servizio".
xinetd è la versione "extended" di inetd, usata attualmente.

Configurazione di inetd + tcpd
inetd ha un unico file di configurazione: inetd.conf, che contiene la configurazione di ogni servizio TCP/IP. inetd.d.

TCP "wrapper" ("incartatore"?): tcpd
Il TCP wrapper è il demone tcpd. è un demone "intermedio" che sta fra l'utente ed i servizi TCP che esso richiede.
Una delle sue funzioni, non svolte da inetd, è spedire al demone "system logger" (syslog) stringhe che spiegano ogni tentativo di connessione TCP.
Inoltre, mentre inetd non è in grado di fare differenze fra gli host dai quali proviene una richiesta di servizio, tcpd può

1 Il "parsing" è il riconoscimento e l'isolamento delle singole "parole" significative contenute nella stringa

filtrare le richieste provenienti dagli utenti e negarle o farle passare in base all'indirizzo IP da cui provengono.
Per stabilire gli indirizzi IP "abilitati" e "proibiti" si usano due file: /etc/hosts.allow per gli host "accreditati", /etc/hosts.-deny per quelli "banditi".
In questi due file è possibile anche specificare per quali protocolli il sistema deve concedere o negare il servizio a specifici host IP. Se l'host è abilitato tcpd fa partire il relativo server, altrimenti nega l'accesso.

Esempio: per lasciar passare tutte le richieste al demone finger ed invece "impacchettare" in tcpd il demone telnet dobbiamo scrivere, in inetd.conf:

```
# non "impacchetta" il demone finger:
finger stream tcp nowait root
/usr/sbin/in.fingerd in.fingerd
#"impacchetta" il demone telnet:
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

Le righe di inetd.conf contengono: servizio, protocollo di trasposto, comportamento del demone "dopo l'uso" (wait = rimane in memoria, no wait = viene scaricato), userid con cui verrà eseguito, demone che viene lanciato immediatamente (tipicamente il demone stesso oppure il "wrapper" tcpd), demone lanciato dopo aver fatto i controlli.
Nell'esempio il demone finger viene lanciato direttamente, senza far uso di tcpd, mentre nel secondo caso (telnet) si lancia tcpd, che provvederà a lanciare in.telnetd solo dopo aver verificato l'autorizzazione in hosts.allow. Le richieste di servizio finger NON saranno inviate al system logger.

Autorizzazioni

tcpd controlla il file /etc/hosts.allow per primo, poi /etc/hosts.deny ed agisce di conseguenza; inoltre fornisce i servizi che non sono specificati in nessuno dei due file.

Formato di /etc/hosts.allow e /etc/hosts.deny

<lista di demoni>**:**<lista di host>[**:**<comando di shell>]

Le liste sono composte da schemi come quelli usati per la ricerca di nomi di file. Gli schemi sono separati da uno spazio (blank).
Il comando di shell potrebbe essere usato per far partire uno script che avverte l'amministratore di eventuali tentativi di attacco al sistema.
Esempi:

```
ALL:ALL@ALL # scritto in /etc/hosts.deny nega l'accesso a tutti i servizi TCP/IP

# Perciò funzionano solo i servizi esplicitamente autorizzati
# in /etc/hosts.allow
ALL:ALL EXCEPT localhost # scritto in /etc/hosts.deny nega l'accesso a tutti i servizi
# TCP/IP dall'esterno del nostro computer.

ALL:ALL EXCEPT LOCAL # scritto in /etc/hosts.deny nega l'accesso
# a tutti i servizi PCP/IP dall'esterno della rete locale

in.ftpd:192.168.0.0/255.255.255.240
# scritto in /etc/hosts.allow autorizza l'accesso FTP
# a tutti gli host della rete IP che hanno indirizzo
# da 192.168.0.1 a 192.168.0.15

in.smtpd:Brontolo Dotto
# scritto in /etc/hosts.allow autorizza l'accesso SMTP
# (scrittura della posta elettronica) agli host Brontolo e Dotto

in.smtpd:monti.it
# scritto in /etc/hosts.allow autorizza l'accesso SMTP
# agli host che appartengono al dominio DNS monti.it

in.telnetd:.binladen.org:(/bin/echo/"Tentativo di accesso telnet da organizzazione so-
spetta, client TCP %c, server TCP %s")&
```

Quest'ultimo comando manda in console un avvertimento che riguarda il tentativo di collegamento.
tcpd ha un linguaggio di comandi, che esiste anche in una forma "estesa", e che comprende le macro %s (server) e %c (client), che espandono stringhe descrittive del client e server IP coinvolti nella "violazione".
Volendo si potrebbe anche mandare in posta elettronica, mettendo il "pipe" con "mail" il risultato del comando echo:

```
in.telnetd:.binladen.org:(/bin/echo/"Stringa come prima" | mail -s "Accesso non autoriz-
zato" root) &
```

è molto interessante la possibilità di indicare utenti di un certo computer Unix invece che host IP.
Se nella <lista di host> compare un'indicazione nel formato <Schema nome utente>@<Schema nome dominio NIS>, tcpd fa un'interrogazione a tutti gli host il cui nome o indirizzo IP corrispondono a <Schema nome dominio NIS>, per

verificare se il nome dell'utente che ha lanciato il processo che richiede il servizio è conforme a quanto indicato dallo <Schema nome utente>.

Configurazione di xinetd

xinetd (extended inetd) è un demone, più "moderno", che fa le funzioni sia di inetd che di tcpd. Funziona in modo analogo ma si configura in modo diverso.

Funzioni

- tutte le funzioni di inetd
- controllo d'accesso basato sull'indirizzo - nome di dominio (eventualmente anche insieme a tcpd)
- controllo d'accesso basato sul tempo
- log degli eventi
- proxi, in combinazione con un programma NAT (IP-masquerading) può sostituire il wrapper tcpd

Per la configurazione di xinetd.conf bisogna usare una sintassi diversa rispetto a quella di inetd.conf. Per aiutare nella conversione esistono i programmi itox (non più aggiornato) e xconv.pl.
Il file xinetd.conf può contenere la configurazione di tutti i servizi, come succedeva per inetd.conf, oppure può avere un diverso file di configurazione per ogni protocollo che il demone gestisce. Questi file sono contenuti in un directory (tipicamente /etc/xinetd.d/). Tutti i file di questo directory vengono "inclusi" in xinetd.conf, uno per ogni servizio IP, TCP, o UDP esistente.
Per effettuare questa inclusione il file xinetd.conf deve contenere il comando:

includedir <nome del directory in cui sono memorizzati i file di configurazione dei servizi>

xinetd.conf contiene diverse "sezioni" una per ogni servizio che controlla. La sezione che sta sempre nel file xinetd.conf è la sezione "defaults", che indica il comportamento del demone nelle condizioni di default, le altre di solito stanno nei file relativi ai singoli servizi.
La sintassi della dichiarazione di una sezione è la seguente:

<Nome della sezione>
{ <configurazione di attributi da usare da parte del servizio> **}**

La sezione di nome "defaults" stabilisce gli attributi di default per tutti i servizi. Questi attributi possono essere "sovrascritti" da quelli definiti nelle singole sezioni specifiche.
Il formato usato all'interno della sezione è il seguente:

<Attributo> <Operatore> <Valore> [<Valore> ..]

<Operatore> può essere:

La stringa "=" per assegnare la stringa <valore> all'attributo
La stringa "+=" per aggiungere la stringa valore alla lista dei valori dell'attributo
La stringa "-=" per togliere la stringa valore dalla lista dei valori dell'attributo

Esempi:

```
no_access = 195.32.14.
only_from -= .binladen.org
```

I principali attributi delle sezioni:

disable: se = yes => il servizio non viene attivato
no_access: stabilisce la lista degli host che non hanno accesso al servizio. Per specificare gli host si possono dare gli indirizzi IP di singoli host, di intere reti o nomi di dominio DNS, eventualmente "semplificati". Usare no_acces in congiunzione con only_from può in qualche caso "strano" far sì che xinetd non sappia cosa fare. Per questo è meglio usare solo only_from, che, se usato senza alcun parametro, non abilita nessuno.
only_from: stabilisce la lista dei client autorizzati
log_on_success: stabilisce cosa scrivere nel file di log quando si inizia un collegamento accettato
log_on_failure: stabilisce cosa scrivere nel file di log quando xinetd rifiuta un collegamento
port: stabilisce il numero di port TCP usato dal servizio (i default sono in /etc/services)
server: la path del demone server, che viene lanciato da xinetd quando la richiesta di collegamento TCP è accettata.
socket_type: il tipo di protocollo usato dal servizio:

stream (protocollo affidabile: TCP)
dgram (protocollo a datagramma: UDP)
raw (nessun protocollo di livello superiore, usa solo datagrammi IP)

wait: se =yes il server è "monoprogrammato": prima di accettare la prossima richiesta di connessione attende che si concluda la precedente.
access_times: stabilisce intervalli di tempo in cui il servizio è abilitato
id: stringa che identifica il servizio nei file di log (è utile quando lo stesso servizio viene fornito in due "versioni" diverse. P.es. potremo avere un server FTP interno ed uno esterno)

Esempio di file di configurazione /etc/xinetd.conf:

```
# alcune definizioni di default:
defaults
{ instances = 60 # n.ro max di server dello stesso tipo contemporanei
log_type = SYSLOG authpriv # usa SYSLOG per il log degli eventi
log_on_success = HOST PID # visualizza indirizzo IP e ID per processo
log_on_failure = HOST # visualizza indirizzo IP
# paranoia: se non si specifica altrimenti,
# sbatto tutti fuori sempre e da subito:
only_from =
# se qui ^ sopra non c'è nessun indirizzo nessuno può accedere
}
# inclusione di tutti i file compresi nel directory indicato:
includedir /etc/xinetd.d
# ^ xinetd considererà tutti i file del directory /etc/xinetd.d come se
# fossero scritti qui sotto.

# fine del file /etc/xinetd.conf
```

Esempio di file di configurazione di un servizio (file /etc/xinetd.d/telnet):

```
# description: The telnet server serves telnet sessions; it uses
# unencrypted username/password pairs for authentication.
service telnet
{
    disable = yes
    flags = REUSE
    socket_type = stream
    wait = no
    user = root
    server = /usr/sbin/in.telnetd
    log_on_failure += USERID
}
```

dato che "disable" è "yes" questo servizio non viene lanciato automaticamente quando il sistema parte, ma solo dietro richiesta esplicita.
L'uso di xinetd è "pulito" e più sicuro, ma può dar luogo ad un decadimento delle prestazioni di rete per quei servizi che richiedono molte connessioni, come tutti i server "importanti" (HTTP, SMNP ..). Per questo di solito i server più "importanti" sono configurato in modo "standalone", non sono presenti fra i demoni configurati in inetd e perciò non vengono da inetd lanciati ad ogni richiesta. Al contrario, partono una sola volta al boot del sistema e rispondono "in proprio" alle richieste di collegamento TCP sui loro port.

2.1 VNC

Virtual Network Console è un protocollo per il controllo remoto dei computer. E' composto da due applicazioni: il server ed il client. Il server (VNC server) esegue sul computer da controllare "da lontano". Il client (VNC viewer) esegue sul computer remoto che controlla quello su cui esegue il server.
Il server:

- attende richieste di collegamenti TCP da parte del programma client. Attende sul suo port standard o su un altro port qualsiasi, stabilito in fase di configurazione
- effettua l'autenticazione dell'utente che vuole collegarsi con il viewer, controllandone la password
- comincia a "catturare" regolarmente le schermate grafiche visibili sul monitor del computer
- comprime la grafica
- spedisce attraverso la rete le schermate compresse

Il client (viewer)

- si collega ed autentica al server
- riceve i dati delle schermate, li decomprime e li visualizza
- spedisce al server i codici dei tasti e gli spostamenti del mouse, così che si possa controllare il computer remoto "come se si fosse lì"

Sviluppato originariamente dai laboratori di ricerca Olivetti in Inghilterra, VNC non è più seguito dal team originale, che non esiste più da tempo, ma è stato portato avanti da molti altri, dato che aveva una licenza libera.
Per esempio esistono "RealVNC", che è anche un prodotto commerciale, "UltraVNC", "TightVNC", che fa una compressione più aggressiva della versione originale, utile nell'uso in reti a banda stretta (es.: accesso Internet con modem acustico).
Il server VNC è "dentro" il codice XWindow di Linux (XOrg86). La versione integrata in XOrg86 è TightVNC (programmi vncviewer, vncserver, vncpasswd , ..).
TightVNC permette l'uso di sessioni crittate SSH, ma solo tra Unix e Unix. Per ora in Windows le sessioni non sono crittate, nè per il server, nè per il client.

2.1.1 Uso di VNC per controllare un computer Linux

Lancio
da terminale remoto telnet (o, molto meglio, SSH) si può lanciare vncserver così:

```
remoto:/# vncserver
```

Se non si dà nessun'altra indicazione, VNC usa il primo display libero, a partire da ":1" (lascia :0 per la sessione X locale del computer su cui gira). Se si vuole che VNC scelga uno specifico numero di display lo si deve specificare esplicitamente; esempio per il display 2:

```
remoto:/# vncserver :2
```

quando parte, vncserver visualizza il numero del display utilizzato e lancia lo script di partenza di X, per l'utente con cui si è fatto il login remoto, per cui ci si può subito collegare con il client, al display indicato da vncserver.
La prima volta che viene lanciato, vncserver chiede una password, che andrà utilizzata per l'accesso da parte del client.

Se si vuole cambiare la password:

```
remoto:/# vncpasswd
```

Visualizzatori (viewer o client VNC)

In Unix si usa vncviewer, programma per Xwindow.

```
locale:/# vncviewer
```

Questa chiamata fa usare il display 0 e richiede all'utente il nome (o l'indirizzo IP) del server.
Se si deve usare un display diverso bisogna indicarlo esplicitamente:

locale:/# vncviewer <host>:<display>

Esempio:
```
# vncviewer 192.169.0.133:2
```

In entrambi i casi verrà richiesta la password; poi si entrerà nel desktop dell'utente che ha lanciato vncserver.

Fra le opzioni di vncviewer
-compresslevel (0-9) per il livello di compressione
-quality per la qualità
Queste opzioni permettono di lavorare anche su reti lente.

In Windows si installa il relativo client (VNC, TightVNC o RealVNC) e le operazioni sono analoghe.

Anche il client TightVNC di Windows permette la configurazione dei livelli di compressione e di qualità.

Una volta conclusa la sessione X, NON verrà conclusa la sessione VNC. Se si vuole concluderla bisogna farlo esplicitamente, usando vncserver o kill:

remoto:/# vncserver -kill :<numero display>

oppure:

remoto:/# kill - term "process ID del server VNC"

Se il server VNC non fa partire il desktop environment desiderato, si può modificare lo script di startup di vncserver, che è diverso per ogni utente e sta all'interno della home directory. Lo script da modificare è <home directory dell'utente>/.VNC/xstartup.
Per esempio, Fedora configura vncserver in modo da far partire il desktop environment twm (che sarà anche leggero e buono per il controllo remoto, ma è proprio bruttino!).
Il file xstartup contiene due righe commentate, da ripristinare togliendo il carattere "#" ad inizio riga. Contiene inoltre le linee che comandano la partenza di twm, da commentare con #, in modo da avere un file fatto così:

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
unset SESSION_MANAGER
exec /etc/X11/xinit/xinitrc

[ -x /etc/VNC/xstartup ] && exec /etc/VNC/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources

#xsetroot -solid grey
#vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
```

In questo modo partirà il destktop environment di default.
Se si vuole far partire uno specifico desktop environment, si può modificare xstartup in modo da introdurre la linea:

```
gnome-session &
```

oppure

```
startkde &
```

In alternativa si può cambiare il desktop environment di default. A questo scopo si può usare switchdesk (con distribuzione Red Hat o compatibile) . Esempio:

```
remoto:/# switchdesk GNOME
```

2.1.2 VNC con SSH

Se si usa VNC da Internet è "consigliato" utilizzarlo attraverso un tunnel SSH, perchè sarebbe facile da attaccare da parte di malintenzionati.
Se il client ed il server sono Unix, per avere un trasferimento dei dati sicuro, perchè crittato, si può creare un "tunnel" SSH, attraverso il quale far passare i dati VNC.
Per farlo bisogna definire un port crittato per VNC sul computer locale, verso il computer remoto, tramite il "potforwarding" SSH:

Su una sessione del terminale del computer da controllare (eventualmente attivata da remoto tramite SSH):

PCdaControllare:/# ssh -C -L 5901:<host remoto>:5901 <utente>@<computer locale>

dove:

<host remoto> è il nome o l'indirizzo IP del computer remoto .
<utente> è il nome utente su <computer locale> (nome del computer locale, anche non completo del dominio)
-C abilita la compressione dei dati (non utile se siamo su una rete locale)
-L (local) specifica il forward SSH sul computer locale, indicando con 5901: di aprire la porta 5901 in ascolto (listen, server TCP), mentre l'ultimo :5901 stabilisce la porta di "uscita" SSH, dal computer locale verso quello remoto.
5901 è in numero di port usato da VNC per default.

Una volta configurato il tunnel SSH, per collegarsi da remoto in modo crittato con VNC, bisogna prima lanciare il server vnc:

```
PCdaControllare:/# vncserver
```

Poi ci si può collegare ad esso così:

```
PCremoto:/# vncviewer
```

in questo modo colleghiamo il viewer alla porta 5901 del computer locale, sulla quale però c'è il "tunnel" verso la porta 5901 del computer remoto, che fornisce il servizio VNC in modo crittato.

Se la rete è protetta da un firewall, si può utilizzare VNC specificando regole che ne permettano l'accesso:

```
firewall:/# iptables -A INPUT -s <host> -p tcp --dport 5901 -j ACCEPT
```

dove <host> è il nome di dominio completo o l'indirizzo IP della macchina locale. Questa regola permette di accedere al port di VNC solo a <host>.

```
firewall:/# iptables -A INPUT -p tcp --dport 5901 -j DROP
```

questa regola esclude l'accesso al port di VNC a tutti gli IP diversi da quello da cui si vuole controllare la macchina remota.