# Setup dell'ambiente virtuale

## 1.1 L'immagine per le macchine virtuali

Il file `sicureti.iso` è l'immagine ISO di una distribuzione "live" di Linux (basata Debian Stretch, ottenuta con `live-build`, `http://live.debian.net`).

Il sistema è configurato con modalità "autologin", qualora dovessero servire le credenziali sono `user:live`.

Il sistema è utilizzabile anche con macchine reali, ma in laboratorio verrà usata con emulatori, in particolare Qemu (ma funziona anche con VirtualBox, VMWare, ecc.).

## 1.2 Qemu

Qemu (`http://qemu.org`) è un emulatore che permette di simulare sistemi hardware completi. È software libero e disponibile per ambienti GNU/Linux, MS Windows, Mac OS X. In ambiente Windows (attenzione: per versioni di Windows a 64 bit, come Windows 7 e 8, è meglio utilizzare una versione di QEmu compilata a 64 bit) è opportuno attivare un X Server, perché sul sistema virtuale non c'è (per risparmiare spazio): in questo modo sarà comunque possibile utilizzare le applicazioni grafiche del sistema virtualizzato. Un X Server facile da installare e usare in ambiente Windows è MobaXterm (`http://mobaxterm.mobatek.net/`).

Nel seguito si farà riferimento alla versione Linux, ma le differenze in altri contesti sono minime e riguardano principalmente il modo di lanciare il programma. I parametri a linea di comando indicati sono disponibili in tutte le versioni. La macchina su cui gira QEmu è detta *host*, la macchina virtualizzata da QEmu è detta *guest*.

Per attivare il sistema virtuale useremo il comando:

```
1 qemu-system-x86_64 -m 768 -enable-kvm \
2  -cdrom sicureti.iso \
3  -net nic,model=virtio \
4  -net user,net=192.168.101.0/24,hostfwd=tcp::8022-:22,hostfwd=tcp::8001-:8000
```

- `qemu-system-x86_64` è il programma di emulazione di un sistema con un processore Intel a 64 bit (`qemu-system-i386` a 32 bit).
- `-m 768` 768 MB di memoria RAM
- `-enable-kvm` (solo Linux) abilitazione di KVM per sfruttare le capacità di virtualizzazione del processore *host*
- `-cdrom sicureti.iso` uso del *file* `sicureti.iso` come *cd-rom*
- `-net nic,model=virtio` interfaccia di rete *hardware* `virtio`

- `-net user,net=192.168.101.0/24,hostfwd=tcp::8022-:22,hostfwd=tcp::8000-:8000` la macchina *guest* usa lo *stack* utente dello *host* (via NAT), il DHCP di QEmu assegnerà un numero IP della rete `192.168.101.0/24`, la porta 22 del *guest* è mappata sulla 8022 dello *host* e la porta 8000 del *guest* è mappata sulla 8000 dello *host* (eventuali altre porte *guest* saranno perciò inaccessibili via *host*).

Se la macchina *host* dispone di sufficiente RAM, si consiglia di aumentare la memoria a disposizione della macchina *guest*.

### 1.2.1 Gestire la macchina virtuale

Agendo con i tasti `Ctrl-Alt-2` mentre è in esecuzione Qemu, si accede al monitor di sistema della macchina *guest;* con `Ctrl-Alt-1` si torna alla macchina *guest.*

Accedere al monitor e provare il comando `help,` ottenendo cosí un elenco delle possibilità. Quando non ci stanno tutte nella finestra corrente, le schermate del monitor possono essere esaminate con `Ctrl-PgUp` e `Ctrl-PgDown`. In modalità monitor con il comando `sendkey` è possibile mandare una combinazioni di tasti alla macchina *guest:* provare `sendkey ctrl-alt-delete`.

Per forzare il salvataggio dei dati sul disco anche se QEmu è stato lanciato in modalità `snapshot`, si può usare il comando `commit all`.

# Tool di base per analizzare lo stack TCP/IP

Per esaminare (e, in alcuni casi, variare) le caratteristiche dello *stack* TCP/IP installato sulla macchina *guest*, si possono utilizzare alcuni tool di base, generalmente presenti in ogni distribuzione di Linux. Prima di iniziare gli esperimenti meglio controllare di riuscire a connettersi all'esterno tramite le applicazioni della macchina guest, per esempio il browser. Le operazioni che seguono richiedono per lo piú i privilegi di amministrazione: il modo piú semplice per iniziare una sessione comandi, da eseguire come `root` user, si apra un terminale e si dia il comando

```
1 sudo -s
```

Dopo di ciò il prompt indica che si sta agendo come `root` (e l'ultimo carattere del prompt è #)

## 2.1 ifconfig

ifconfig è il comando per configurare le *interfacce* di rete.

```
1 ifconfig -a
```

Serve a mostrare tutte (all) le interfacce configurate nel sistema. Un'interfaccia è attiva quando è `UP`. Affinché sia utilizzabile in modalità TCP/IP deve avere (almeno[1]) un numero IP.

Le interfacce attive nella macchina virtuale sono `eth0`[2] e il *local loopback* `lo`.

Per cambiare il numero IP di `eth0`

```
1 ifconfig eth0 10.0.2.16
```

Gli esperimenti possono alterare lo stack rendendolo non funzionante. Potete sempre riavviare la macchina virtuale per ricominciare. In molti casi basta però riavviare semplicemente la configurazione delle interfacce. Nei sistemi della famiglia Debian ciò avviene tramite gli script `ifup` e `ifdown`. Per ripristinare

```
1 ifdown eth0
2 ifup eth0
```

### 2.1.1 Esercizi

- Leggere il manuale di `ifconfig` (man ifconfig)
- Rilevare la netmask di `eth0`

[1] È possibile associare piú di un numero IP alla stessa interfaccia fisica

[2] in realtà dipende dall'interazione fra emulatore e `udev`: potrebbe chiamarsi in modo diverso, per esempio `ens3`. In questo caso sostituire con il nome giusto, dopo aver verificato con `ifconfig -a` o `ip link show`

- Cosa succede se si imposta un numero IP su una sottorete diversa? La connessione con l'esterno continua a funzionare?
- Cambiare il numero MAC dell'interfaccia `eth0` in `00:01:02:03:04:05`

## 2.2 iproute

`ifconfig` è un programma classico negli ambienti Unix (in effetti anche altri sistemi, come Windows hanno ereditato lo stesso nome). Negli ultimi anni, però, si è diffuso un insiemi di programmi che permetto di manipolare tutti i livelli dello *stack* in maniera piú "regolare": `iproute`.

Per vedere le interfacce di rete (livello link)

```
1 ip link
```

e poi

```
1 ip link set dev eth0 up
```

Livello rete

```
1 ip addr
```

e poi

```
1 ip addr change 10.0.0.1 dev eth0
```

Per la sintassi

```
1 ip help
```

e poi per esempio,

```
1 ip link help
```

## 2.3 netstat

Il dispatching dei pacchetti è regolato dalla *routing table*, che può essere manipolata tramite il comando `route`.

Con `netstat` si può controllare lo stato della tabella di *routing*.

```
1 netstat -nr
```

Il `-n` evita di coinvolgere il risolutore DNS nell'operazione, rendendola piú veloce e affidabile (la risoluzione richiede infatti probabilmente la generazione di pacchetti).

Oppure

```
1  ip route
```

`netstat` permette anche di esaminare lo stato delle connessioni in corso o in ascolto

```
1 netstat -taun
```

- Leggere il manuale di `netstat` e dar conto dei parametri usati negli esempi

- Quale processo è in ascolto sulla porta TCP 22?
- Dalla macchina *host* aprire una sessione SSH (ssh −p 8022 mininet@localhost), controllare lo stato delle connessioni dopo questa operazione.

## 2.4 Netcat

Netcat (`http://nc110.sourceforge.net/`) è un programma che permette di stabilire comunicazioni TCP e UDP senza alcun livello applicativo: fornisce sostanzialmente il servizio di creazione di un socket a linea di comando.

L'uso di base prevede che il server si metta in ascolto (−l listen) su di una porta (−p): per quelle $< 1024$ servono i privilegi di root. La porta non deve naturalmente essere già in uso da un altro programma.

```
1 nc −l −p 8000
```

Il client si connetterà con un parallelo

```
1 nc 192.168.1.100 12345
```

Cosí però non è molto utile: bisogna far viaggiare dei dati sulla connessione.

```
1 echo ciao | nc 192.168.1.100 12345
```

In questo caso il server vede il messaggio "ciao". Vale anche nell'altro senso.

```
1 echo ciao | nc −l −p 12345
```

Il client riceverà il messaggio al momento del collegamento.

### 2.4.1 Esercizi

- Leggere il manuale di netcat
- Scambiare il messaggio 'ciao' con netcat tra la macchina *guest* e *host*