QEMU

dr. Andrea E. Naimoli

Università degli Studi di Trento
Dipartimento di Ingegneria e Scienza dell'Informazione
via Sommarive 14
I - 38050 Trento - Povo, Italy

- Qemu
- Debian GNU/LINUX
- Configurazione e compilazione del kernel

Esercitazione 2

Obiettivo: utilizzare un computer virtuale che esegue il sistema operativo GNU/Linux e dotarlo di un nuovo kernel

Ingredienti:

- qemu: emulatore di processore con virtualizzazione completa
- Debian GNU/Linux: una distribuzione GNU/Linux
- kernel Linux (*provare da soli*)
  - ✓ configurazione
  - ✓ compilazione
  - ✓ installazione e utilizzo

ma prima di tutto...

verifichiamo di avere **TUTTI** una nostra directory in /sysop/ :

```
cd /sysop/<login>
```

Se non si potesse sfruttare, attenzione alla quota!

Qemu (1/2)

Qemu e':

- un **emulatore di processore**: x86 (32 e 64 bit), PowerPC, (32 e 64 bit), Sparc (32 e 64 bit), MIPS, ARM e m68k
- e' **multipiattaforma**: funziona su x86 (32 e 64 bit), PowerPC, Sparc (32 e 64 bit), ARM, Alpha, S390, m68k e ia64
- emula numerose periferiche (scheda video, di rete ecc.)
- e' Software Libero (licenza GPL)
- e' veloce
- e' in forte sviluppo

Qemu (2/2)

- Cosa posso fare con Qemu?
  - ✓ Eseguire un altro SO (ospite) all'interno di quello in esecuzione (ospitante), es.:
    - GNU/Linux dentro GNU/Linux
    - MS Windows dentro GNU/Linux
    - GNU/Linux dentro MS Windows
  - ✓ eseguire un SO per un'archiettura (ospite) dentro un altro SO su un'altra archiettura (ospitante), es.:
    - MS Windows per x86 dentro GNU/Linux per PowerPC
  - ✓ Eseguire programmi per GNU/Linux o OSX di un'architettura su un'atra architettura
- non richiede permessi speciali per essere eseguito

perche' utilizzare Qemu?

- Per il corso utilizzeremo GNU/Linux e avremo necessita' di sostituire il kernel e introdurre moduli
- Ogni sistema operativo richiede una partizione del disco apposita...
- ...emuliamo invece un intero computer via software! Il suo disco sara' un file sul nostro hard disk
- Possiamo far convivere sistemi operativi differenti in contemporanea

Qemu: dettagli tecnici

- Periferiche emulate:
  - scheda grafica Cirrus PCI VGA
  - mouse e tastiera PS/2
  - supporto cdrom
  - floppy disk
  - scheda di rete NE2000 PCI e RTL 8139
  - porte seriali
  - 2 schede audio (Creative SoundBlaster, Ensoniq 1370)
  - porta USB
- Supporto per SMP

qemu: utilizzo pratico

- Creazione di un disco virtuale (1800MB) su file:

  qemu-img create -f qcow2 /tmp/disco.img 1800M

- Installazione di un sistema operativo da CD:

  qemu -hda /tmp/disco.img
  -cdrom /dev/cdrom
  -boot d
  qemu -hda /tmp/disco.img
  -cdrom /sysop/_shared/debian-500-i386-netinst.iso
  -boot d

- Esecuzione di un sistema operativo gia' installato su un disco virtuale:

  qemu /tmp/disco.img

- Modalita' fullscreen: Ctrl-Alt-f

qemu: utilizzo pratico in aula

- È stato predisposto un file/disco virtuale **comune** con **Debian GNU/Linux 6.0.2 (squeeze)** gia' installato e con alcune personalizzazioni apposite per il corso
- Attenzione alla quota: avete solo 50Mb di spazio sulla vostra home directory! Utilizzate se possibile la directory /sysop/*<login>*
- Usare eventualmente l'alias:
  alias qemu="qemu-system-i386"
- Come utilizzare lo stesso disco con tanti studenti?
  - ✓ Modalita' **snapshot**: **non** modifica il disco originale:
    qemu -snapshot
    -hda /sysop/_shared/img/disk.img
  - ✓ oppure con un **disco delle differenze**, personale:
    - creazione del disco delle differenze:
      qemu-img create -b /sysop/_shared/img/disk.img
      -f qcow /sysop/<login>/diskdiff.img
    - avvio qemu utilizzando il nuovo disco personale:
      qemu -hda /sysop/<login>/diskdiff.img

Qemu e GNU/Linux

- *qemu -hda ./diskdiff.img*
  -boot c -localtime –usb
  -m 975
  -redir tcp:2222::22
- Alla schermata di login utilizzare
  - ✓ utente **root**, password: **root**
- Potete creare altri utenti se volete
- Per terminare eseguire: **shutdown -h now o halt**
- Alla fine potete chiudere la finestra di qemu
- Se interrompete il sistema prima della sua normale chiusura potete ottenere un filesystem inconsistente

Qemu e la rete

- Qemu permette l'utilizzo della rete al sistema ospite (guest)
- Le operazioni permesse sono quelle accessibili all'utente che esegue qemu
- Il sistema ospite risiede in una sottorete (virtuale) del sistema ospitante e puo' collegarsi all'esterno
- Ping NON funziona (richiede privilegi)
- E' possibile redirigere porte dall'ospitante (host) all'ospite (guest)

qemu -redir tcp:2222::22 ...

(questo esempio rimappa la porta 22 virtuale sulla 2222 reale)

Debian GNU/Linux

- Debian e' un progetto basato sullo sviluppo di un sistema operativo libero a partire dal progetto GNU
- Ha come prodotto principale la distribuzione Debian GNU/Linux
- Dispone di 15k pacchetti software
- Funziona su: i386, x86-64, PowerPC, 68k, SPARC, DEC Alpha, ARM, MIPS, HPPA, S390 e IA-64
- E' la base di molte distribuzioni, es.: Ubuntu, Knoppix, ...
- E’ nota per il sistema di gestione dei pacchetti APT e per la qualita' dei propri prodotti

debian

APT

- APT: Advanced Package Management

  apt-get, apt-cache, dpkg, aptitude,...

- Per cercare i pacchetti contenenti il nome *nano*

  apt-cache search nano

- per ottenere informazioni dettagliate sul pacchetto *nano*

  apt-cache show nano

- per aggiornare i repository dei pacchetti

  apt-get update

- per installare il pacchetto nano ed eventualmente tutte le sue componenti necessarie (dipendenze):

  apt-get install nano

- per rimuovere il pacchetto nano installato prima:

  apt-get remove nano

SSH – client (1/2)

- SSH: Secure Shell e' un insieme di standard e protocolli per stabilire connessioni sicure con un computer remoto
- L'architettura di SSH e' client-server (server: porta 22)
- OpenSSH e' un'implementazione libera (la piu' diffusa)
- Client e server sono gia' installati di solito, altrimenti:

apt-get install openssh-server

apt-get install openssh-client

apt-get install ssh

- Per collegarci a un computer che ha un SSH-server:

ssh <nome_computer>

SSH – client (2/2)

- Per collegarci alla macchina virtuale:

  ssh -p 2222 root@localhost

  (nota: la porta 2222 e' quella indicata in “qemu -redir...”)

- Per copiare un file su un computer remoto

  scp <file> <login>@<nome_computer>:/directory/

- Per copiare un file sulla macchina virtuale, nella home dell'utente:

  scp -P 2222 <file> root@localhost:/home/

- Per copiare un albero di directory sulla macchina virtuale:

  scp -P 2222 -pr <directory> root@localhost:/home/so1/

  scp -P 2222 root@localhost:/home/<file> <file>

Kernel Linux: introduzione

- **kernel**: nucleo del sistema operativo che fornisce accesso sicuro, controllato e semplificato all'HW ai processi in esecuzione
- **Linux**: un kernel
  - ✓ il piu' flessibile (funziona dai cellulari ai supercomputer)
  - ✓ 7 milioni di righe di codice
  - ✓ software libero (licenza GPL v2)
  - ✓ versione attuale: 3.2.4
  - ✓ che versione sto utilizzando?

```
uname -a
Linux debian 2.6.26-2-686 #1 SMP Wed Feb 10
08:59:21 UTC 2010 i686 GNU/Linux
```

Un kernel Linux su misura

- perche' un kernel su misura?
  - ✓ maggiori performance
  - ✓ e' “vestito su misura” per il nostro HW
  - ✓ necessario in molti casi (embedded e high-end)
  - ✓ imparare a crearlo e' utile per avvicinarsi a studiarlo
- strumenti per creare un kernel su misura (fino a v2.6.18):
  - ✓ codice sorgente del kernel : *http://www.kernel.org*
  - ✓ compilatore: gcc (v3.2 o successiva)
    gcc --version
  - ✓ linker (binutils): strumenti di manipolazione di object files
    ld -v (v2.12 o successiva)
  - ✓ make v3.79.1 o successiva: *make --version*

Codice sorgente del kernel

(*da provare a casa*)

- Il codice sorgente lo trovate qui: *http://www.kernel.org*
- Create la directory /tmp/linux (sulla macchina host)

  mkdir /root/kernel_src
- Copiate li'

  cp linux-3.2.4.tar.bz2 /root/linux
- Copiate il kernel compresso nella macchina virtuale qemu

  scp -P 2222

  *   /tmp/linux/linux-3.2.4.tar.bz2 root@localhost:/usr/src*
- Collegatevi alla macchina virtuale con ssh

  ssh -p 2222 root@localhost
- Decomprimete il pacchetto:

  cd /usr/src

  tar jxvf linux-3.2.4.tar.bz2

Configurazione del kernel

(*da provare a casa*)

- Verifica:

  ls
  linux-3.2.4
  linux-3.2.4.tar.bz2

- Creiamo il link simbolico del tipo

ln -s /usr/src/linux-3.2.4 linux

cd linux

- Configurazione del kernel con parametri di default:

make defconfig

- Guardiamo il file .config generato
- Configurazione del kernel con menu interattivo:

make menuconfig (or xconfig or gconfig)

Compilazione del kernel

(*da provare a casa*)

- make... e aspettiamo 30-50 minuti
- Alcune opzioni avanzate di compilazione:
  - multithreading: *make -j4 , make -j8 , make -j*
  - una sottodirectory: *make M=drivers/usb/serial*
    (rilanciare poi make per fare il linking con il kernel creato)
  - un file specifico: *make drivers/usb/serial/visor.ko*
  - output in un'altra directory:
    make O=/altra/directory/
  - ripulire: *make mrproper*
  - compilare per un'altra architettura:
    make ARCH=x86_64 defconfig
  - compilazione distribuita: *make -j16 CC="distcc"*

Installazione del kernel

(*da provare a casa*)

- installiamo i moduli kernel creati:
  `make modules_install`
  (viene creata `/lib/modules/linux-3.2.4`)
- installare l'immagine del kernel creata:
  `make install`

Cosa fa *'make install'*?

- Verifica che il kernel sia stato compilato con successo
- Copia il kernel statico in */boot* e lo rinomina opportunamente
  cp arch/i386/boot/bzImage /boot/bzImage-linux-2.6.31.12
- Copia la nuova tabella dei simboli in */boot* e la rinomina
  cp System.map /boot/System.map-linux-2.6.31.12
- Opzionalmente copia .config in /boot e lo rinomina
  cp .config /boot/config-linux-2.6.31.12
- Opzionalmente crea un ramdisk con i moduli necessari
- Aggiorna il bootloader

Aggiornamento del bootloader

(*da provare a casa*)

- **bootloader**, o meglio **second-stage bootloader**: programma che determina quale kernel far partire all'accensione del computer

- Bootloader per GNU/Linux:

  GNU GRUB, Lilo, Syslinux...

  update-grub

- Reboot e all'avvio scegliere il nuovo kernel

- Provare per credere: uname -a

initrd / initramfs

(*da provare a casa*)

- **initrd**: initial ramdisk, e' un file contenente un piccolo filesystem contenente alcuni moduli del kernel e le istruzioni per caricarli.
- Esempio di dilemma: il kernel deve accedere a dispositivi inusuali per poter leggere le directory di sistema contenenti i moduli kernel per accedere a questi dispositivi.
- Soluzione: mettere i moduli necessari anche in initrd
- Per costruire un file initrd:
  - ✓ mkinitrd
  - ✓ mkinitramfs

Esercizi per casa

- Mettere in pratica quanto indicato nel capitolo 6 e 7 del libro *Linux Kernel in a Nutshell* (vedi riferimenti)
- Compilare e installare l'ultimo kernel Linux scaricabile dal sito http://www.kernel.org

Riferimenti

- qemu, *http://fabrice.bellard.free.fr/qemu/*
- Progetto Debian, *http://www.debian.org*
- APT, Advanced Package Tool: *http://www.debian.org/doc/manuals/apt-howto/index.it.html*
- OpenSSH, Secure Shell:

  http://a2.swlibero.org/a21.htm
- Linux Kernel in a Nutshell, di Greg Kroah-Hartman, O'Reilly http://www.kroah.com/lkn/ (disponibile gratuitamente sotto licenza Creative Commons)