

Gentoo GNU/Linux on usb device

Giuseppe `lan` Marocchio

info@giuseppe-marocchio.com

Perchè Gentoo?

- 1• Assenza di un programma di installazione**
- 2• Ottima documentazione (anche in italiano)**
- 3• Ottimo tool per la compilazione dei pacchetti**
- 4• Root ha il potere totale sul sistema**
- 5• Totalmente personalizzabile**

Cosa Ci serve?

- 1• Una chiaveta usb (minimo 128 mb)**
- 2• Un sistema linux**
- 3• Il tarball dello stage 1 di Gentoo**
- 4• Una connessione Internet veloce oppure:**
- 5• Sorgenti dei pacchetti**
- 6• Il tarball contenente il portage (elenco dei pacchetti disponibili con le istruzioni per compilarli)**

Iniziamo:

1• Creiamo per prima cosa una directory di lavoro che conterrà il nostro piccolo sistema gentoo mentre viene alla luce:

```
jasmine / # mkdir /usbsys
```

2• Scarichiamo lo stage1 di gentoo aggiornato:

```
jasmine / # wget http://gentoo.osuosl.org/releases/x86/2004.2/stages/x86/stage1-x86-2004.2.tar.bz2
```

3• Scarichiamo md5 dello stage e verifichiamo l'integrità dello stage:

```
jasmine / # wget http://gentoo.osuosl.org/releases/x86/2004.2/stages/x86/stage1-x86-2004.2.tar.bz2
```

```
jasmine / #cat stage1-x86-2004.2.tar.bz2.md5
```

```
327f1ef412a9c4e5aad7385e721b16bb *stage1-x86-2004.2.tar.bz2
```

```
jasmine / #md5sum stage1-x86-2004.2.tar.bz2
```

```
327f1ef412a9c4e5aad7385e721b16bb stage1-x86-2004.2.tar.bz2
```

Bene Possiamo Continuare!

Iniziamo L'installazione:

1•Decomprimiamo lo stage 1 nella directory di lavoro:

```
jasmine / # tar -jxvpf stage1-x86-2004.2.tar.bz2 -C /usbsys/
```

2• Vediamo se siamo sulla giusta strada:

```
jasmine usbsys # ls -alh /usbsys/
```

```
total 40K
```

```
drwxr-xr-x 17 root root 408 Jul 18 01:15 .
```

```
drwxr-xr-x 23 root root 680 Oct 24 21:06 ..
```

```
drwxr-xr-x 2 root root 2.4K Jul 18 02:01 bin
```

```
drwxr-xr-x 2 root root 96 Jul 18 01:15 boot
```

```
drwxr-xr-x 9 root root 35K Jul 18 01:15 dev
```

```
drwxr-xr-x 16 root root 1.4K Jul 18 02:02 etc
```

```
drwxr-xr-x 2 root root 72 Jul 18 01:15 home
```

```
drwxr-xr-x 5 root root 1.9K Jul 18 02:02 lib
```

```
drwxr-xr-x 4 root root 120 Jul 18 01:15 mnt
```

```
drwxr-xr-x 2 root root 72 Jul 18 01:15 opt
```

```
drwxr-xr-x 2 root root 72 Jul 18 01:15 proc
```

```
drwx----- 2 root root 72 Jul 18 01:15 root
```

```
drwxr-xr-x 2 root root 472 Jul 18 01:54 sbin
```

```
drwxr-xr-x 2 root root 72 Jul 18 01:15 sys
```

```
drwxrwxrwt 2 root root 72 Jul 18 02:02 tmp
```

```
drwxr-xr-x 11 root root 400 Jul 18 02:02 usr
```

```
drwxr-xr-x 11 root root 288 Jul 18 01:15 var
```

Scegliamo L`ottimizzazione:

Esistono molte ottimizzazioni per GCC es:

`CFLAGS="-march=athlon-xp -O3 -pipe -fomit-frame-pointer "` (molto "spinta" il mio pc di casa)

Più ottimizziamo i binari per una architettura specifica più saranno grandi e di conseguenza pesanti da caricare e compilare. Nel nostro caso è preferibile compilare per X86 in modo da avere binari compatibili e piccoli.

-Editiamo `/usbsys/etc/make.conf` modificando il valore `CFLAGS` nel seguente modo:

`CFLAGS="-mcpu=i386 -Os -pipe"`

Chroot e Preparazione

E` giunto il momento di mettere online l'embrione di sistema per procedere all'installazione, è sufficiente impostare i dns, con un semplice:

```
jasmine usbsys # cp /etc/resolv.conf /usbsys/etc/resolv.conf
```

-Montiamo il file system virtuale proc:

```
jasmine usbsys # mount -t proc proc /usbsys/proc/
```

-Entriamo nel sistema tramite Chroot:

```
jasmine usbsys #chroot /usbsys/ /bin/bash
```

```
bash-2.05b#
```

-Perfetto siamo dietro! È il momento di scaricare la lista dei pacchetti tramite Rsync:

```
bash-2.05b# emerge sync
```

-Bene attendiamo qualche minuto (a seconda della connessione)

Inizio dell'installazione

E` giunto il momento di iniziare a installare in questa prima fase ricompileremo gcc e i pacchetti di base per il sistema:

```
bash-2.05b# cd /usr/portage/  
bash-2.05b# ./scripts/bootstrap.sh
```

**Gentoo Linux; <http://www.gentoo.org/>
Copyright 1999-2004 Gentoo Foundation; Distributed under the GPLv2
Starting Bootstrap of base system ...**

Eccoci Qua a compilare ora possiamo pure dedicarci ad altro finchè aspettiamo.... (consiglio la lettura degli howto di www.gentoo.org)

Base minima di installato

Ora andremo a installare ciò che ci serve:

-Se abbiamo una chiavetta da 256 mb o meno adremmo subito a istallare ciò che ci serve.

Esempio:

```
emerge net-misc/dhcpd
emerge vim
emerge links
emerge irssi
emerge reiserfsprogs
emerge xfsprogs
emerge <QuelloCheVuoi>
```

Per cercare un pacchetto si usi l'opzione emerge -s (--search), per visualizzare le dipendenze da installare si usi l'opzione -p (--pretend). Un bell emerge -help non guasta mai per vedere le innumerevoli funzionalità di emerge

Chiavetta usb \geq 500mb

Nel caso di una chiavetta di queste dimensioni è possibile, anzi consigliabile installare un gentoo completa.

Azitutto subito dopo finito il bootstrapping diamo un bel:

`emerge system`

Questo comando provvederà a installare tutti i pacchetti indispensabili per il buon funzionamento del sistema, dopodichè possiamo installare in tutta tranquillità ciò che ci serve.

Esempio:

```
emerge x11-xorg #il fork del famoso server Xfree
emerge mozilla-firefox #il miglior borwser esistente
emerge fluxbox #un semplice window manager
```

Il kernel e il metodo di boot

Abbiamo 3 opzioni che possiamo scegliere:

- Boot da floppy con kernel specifico**
- Boot da usb con kernel generico**
- Boot da usb con kernel specifico**

Un kernel specifico per un sistema può stare su un floppy, un kernel generico (che andremmo a creare con genkernel) necessita dell'initrd e di una bzImage così grande che non può stare su un floppy.

È possibile usare tutti e tre i sistemi ma necessariamente se ci spostiamo su un pc diverso dal nostro dovremmo fare il boot da usb per usare il kernel generico.

Compilare con Genkernel

Installiamo genkernel:

`emerge genkernel`

Ora che lo abbiamo possiamo installare i sorgenti del kernel:

`emerge -s sources`

Con questo comando avrete un'idea degli innumerevoli kernel patchati per i vari usi. Per un sistema 686 consiglio i gentoo-dev sources:

`emerge gentoo-dev-sources`

per procedere alla compilazione automatica diamo:

`genkernel all`

e a fine del processo troveremo nella nostra /boot tutti i file che ci serviranno, da notare che genkernel ha bisogno di particolari parametri per il boot (si veda messaggio a fine utilizzo)

Compilare il kernel manualmente(1):

per chi non fosse esperto, i sorgenti del kernel stanno in /usr/src/linux, per entrare nel “setup di configurazione” spostiamoci nella directory che contiene i sorgenti e digitamo make menuconfig.

Ecco cosa ci serve sicuramente:

Loadable module support --->

- Enable loadable module support**
- Module unloading**
- Forced module unloading (NEW)**
- Automatic kernel module loading**

I moduli ci servono per gestire parti dinamiche del kernel

Compilare il kernel manualmente:

Device Drivers --->

Block devices --->

RAM disk support

(4096) Default RAM disk size (kbytes) (NEW)

Initial RAM disk (initrd) support

Support for Large Block Devices

ATA/ATAPI/MFM/RLL support --->

VIA82CXXX chipset support

SCSI device support --->

SCSI disk support

SCSI generic support

USB support --->

USB Mass Storage support

File systems --->

Second extended fs support

Ext3 journalling file system support

Ext3 extended attributes

Reiserfs support

XFS filesystem support

Pseudo filesystems --->

/proc file system support

/dev file system support (OBSOLETE)

/dev/pts Extended Attributes

Virtual memory file system support (former shm fs)

HugeTLB file system support

Operazioni finali sul sistema

Per il buon funzionamento della nostra gentoo abbiamo bisogno di 3 pacchetti importantissimi per la gestione dei device:

`emerge coldplug hotplug udev`

ora aggiungiamo coldplug e hotplug al runlevel di default

`rc-update add hotplug default`

`rc-update add coldplug default`

impostiamo la password di root del sistema con un canonico

`passwd`

creiamo eventuali utenti con il comando adduser es:

`adduser -G users,audio,wheel cicciopasticcio`

e impostiamo la password con un:

`passwd cicciopasticcio`

Boot loader

Il bootloader è la parte più importante. Ecco come mettere lilo nell'mbr della penna usb emerge lilo

```
lba32
boot = /dev/sda #dove verrà installato lilo
map = /boot/.map
prompt #chiede la scelta
timeout= 15 # tempo massimo
default = gentoo usb
#in caso di genkernel
image = /boot/bzImage
        root=/dev/ram0
        label = gentoo usb Genkernel
        read-only
        append="real_root=/dev/sda2"
        initrd= /boot/initrd
#in caso di un kernel fatto su misura
image = /boot/bzImage
        root=/dev/sda2
        label = gentoo usb
        read-only
```

e ora diamo il comando lilo per installarlo

Bootloader su floppy

**inseriamo un floppy e diamo mkfs.ext2
/dev/fd0, montiamolo e copiamo il kernel nel
floppy**

```
lba32
boot = /dev/fdo #dove verrà installato lilo
map = /boot/.map
prompt #chiede la scelta
timeout= 15 # tempo massimo
default = gentoo usb
#in caso di un kernel fatto su misura
image = /boot/bzImage
        root=/dev/sda2
        label = gentoo usb
        read-only
```

e ora diamo il comando lilo per installarlo

Grub il cugino di lilo (mica tanto)

GRUB è un boot loader multipiattaforma estremamente flessibile e potente. Ha un propria CLI in cui inserire a mano i parametri di boot o può presentare un'interfaccia a menu configurabile tramite il file /etc/grub.conf.

Per installare grub sul settore di avvio basta dare il comando: `grub-install /dev/hda` (o altro nome di device valido) A differenza di LILO non c'è bisogno di ridare il comando ogni volta che si cambia la configurazione.

`Default=0` #Imposta come default la prima "label" sotto indicata

`timeout=10` #Imposta a 10 secondi il tempo di attesa prima di caricare automaticamente l'entry di default.

`password --md5 $1$6dòüZßXÈ$bXTLL8IbDhnwmjyaNNcPG.` #Imposta una password, criptata, da fornire per poter accedere al menu o alla command-line.

`title Red Hat Linux (2.4.7-10)` #Il titolo della prima scelta del menu ("label")

`root (hdo,2)` #Hard disk (primary master) e partizione (terza) del device di root.

`kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda3` #Il path per il kernel

`initrd /boot/initrd-2.4.7-10.img` #Il path per il file system da montare su Ram Disk al boot.

Operazioni finali

-Settare /etc/fstab secondo le proprie esigenze.

-Cancellare i file inutili:

```
rm -r /usr/src/*
```

```
rm -r /usr/share/man
```

```
rm -r /usr/share/doc
```

```
rm -r /usr/portage
```

```
rm -r /root/*
```

```
rm -r /var/tmp/*
```

```
rm -r /tmp/*
```

Idee....

Qualche spunto per future rielaborazioni...

- Creare un file system compresso con squashfs**
- Montare directory via rete (smb, nfs)**
- Installare xorg e levare driver e altri file inutili.**
- Utilizzare altre distribuzioni più leggere**
- Crearsi da zero una distribuzione dedicata allo scopo**

Utilità:

- Workstation diskless in internet caffè, scuole, biblioteche.**
- Utilizzare l'accoppiata xorg+rdesktop per stazioni di lavoro in azienda con terminal server windows**
- Disaster recovery :)**
- Per avere il proprio ufficio sempre nel taschino della camicia**