Compilazione Kernel 2.6x con supporto BootSplash

HowTo scritto da: Ed3n
E-Mail: ed3nuzzo[at]gmail[dot]com
WebSite: www.linux.le.it
Licenza: Gnu Fdl
Sistema Operativo: Debian Gnu/Linux Sid

---

1)Introduzione

La compilazione del kernel è uno degli atti più complessi, ma anche uno dei più interessanti, quando utilizziamo un sistema operativo Gnu/Linu. Come spero molti di voi sanno, la compilazione di un Kernel Linux è un operazione molto delicata, ma al contempo molto importante, in quanto permette all'utente di gestire la propria macchina come meglio crede, magari abilitando il supporto a particolari periferiche, disabilitando driver(aka moduli) che non servono al funzionamento del nostro hardware, aggiungendo delle nuove features che rendono il sistema più performante, ma anche più gradevole, sopratutto nell'aspetto. Proprio di quest'ultima caratteristica ci occuperemo in questo HowTo, che ha come compito quello di spiegare come compilare un kernel della serie 2.6x, patchare il kernel con una patch nota come Nitro2, configurare il kernel in modo da abilitare il supporto al BootSplash e VesaFrame Buffer, applicare un immagine(.jpg) di sfondo all'immagine del kernel, installare il programma per l'applicazione dell'immagine ed infine, configurare LiLo per il nuovo Kernel. Se tutto va bene, ad operazione(i) finita, dovreste avere il boot della vostra distro con una bella immagine di sfondo a farvi compagnia :) Infine, voglio ricordarvi, che le mie prove sono state fatte su un sistema operativo Debian Gnu/Linux Unstable(sid), quindi questa guida si baserà unicamente su questo sistema. Non sò come vanno le cose con altre distro, ma dovrebbero essere quasi le stesse.

2)Cosa serve?

- Sorgenti del kernel 2.6x
- Patch Nitro per il kernel 2.6.10 reperibile qui: http://sepi.be/nitro.php
- SplashUtils, potete scaricarlo con apt
- Sfondi per il boot reperibili su: http://www.bootsplash.de/
- Mooolta pazienza... :)

3)Cominciamo

Come prima cosa, procuriamoci i sorgenti di un kernel della serie 2.6x. Potete scaricare la tarball dei

sorgenti dal sito: www.kernel.org. Nelle mie prove ho compilato i sorgenti di un kernel “Vanilla”, anche perchè, non so in che cosa andrei in contro se usassi i sorgenti che forniscono i repositories di Debian. Ho preferito utilizzare un bel kernel liscio liscio senza patch per poi mettere su la nitro. Una volta scaricati i sorgenti, copiateli nella cartella che di solito usate per la compilazione, estraeteli, e create il solito link simbolico a /usr/src/linux:

```
filth:/usr/src# tar -xjvf linux-2.6.10.tar.bz2
filth:/usr/src# ln -s linux-2.6.10 linux
```

Dopo aver fatto ciò, scaricate dal sito http://sepi.be/nitro.php la patch nitro. Al momento della scrittura di quest'howto la release della patch è arrivata alla Nitro4. Io personalmente non l'ho ancora provata, ma non credo ci siano grossi cambiamenti dalla release che ho usato io nelle mie prove. Dopo aver scaricato la patch, create una cartella sempre in usr/src/ col nome che volete, estraete la patch, e copiatela nella directory che avete appena creato:

```
filth:/usr/src# mkdir -p patch
filth:/usr/src# cp /home/user/patch-2.6.10-nitro2.bz2 /usr/src/patch
filth:/usr/src# cd patch
filth:/usr/src/patch# bunzip2 patch-2.6.10-nitro2.bz2
```

Ora è arrivato il momento di patchare il nostro bel kernel vanilla con la nitro2. Spostatevi nel link simbolico della cartella dei sorgenti del kernel ed eseguite questi comandi:

```
filth:/usr/src# cd linux
filth:/usr/src/linux# patch -p1 -E < ../patch/patch-2.6.10-nitro2
```

E aspettate che il processo sia finito. Se sulla vostra macchina non avete ancora installato il programma patch, le ncurses e tutto ciò che serve per la compilazione di un kernel della serie 2.6x, date questi comandi:

```
filth:/usr/src/linux# apt-get install patch initrd-tools cramfsprogs dash gcc make binutils g++ cpp libncurses5-dev module-init-tools
```

Adesso prepariamoci alla configurazione del kernel.

4)Configurazione

Eccoci giunti al momento più “difficile” dell'operazione. Adesso dobbiamo configurare il kernel. Il metoto che preferisco io, è quello di operare con il menù scritto con le librerie ncurses di cui sopra. Il comando da dare è questo:

```
filth:/usr/src/linux# make menuconfig
```

Una volta entrato nel menù di configurazione, dopo aver abilitato i moduli di cui avrete bisogno per la vostra macchina, assicuratevi di abilitare il supporto STATICO nel kernel di:

```
Block devices > [*] Initial RAM disk (initrd) support
Graphics support > [*] Support for frame buffer devices
Graphics support > [*] Support for the framebuffer splash
Graphics support > <*>   VESA VGA graphics support
Graphics support > Console display driver support > [*] Framebuffer Console support
```

5)Compilazione

Per la compilazione del kernel, preferisco procedere alla maniera classica. Per chi usa un sistema Debian c'è un modo particolare per compilare un kernel, detto “DebianWay”. Se vi interessa, cercate delle info su google. Ma passiamo al dunque..anzi, ai comandi:

```
filth:/usr/src/linux# make -j4
```

Questo comando apre quattro thread per la compilazione dei sorgenti, ma non crea ancora nessuna immagine. Prima di creare l'immagine di boot del kernel, assicuriamoci di fare il backup di questo file:

```
filth:/usr/src/linux# cp usr/initramfs_data.cpio.gz usr/initramfs_data.cpio-old.gz
```

Adesso installate il programma che ha il compito di applicare il tema di sfondo al nostro boot con questo comando:

```
filth:/usr/src/linux# apt-get splashutils
```

Create la cartella che deve ospitare il tema che scaricherete da www.bootsplash.de , estraete il tema e applicatelo, attraverso splashutils, al kernel:

```
filth:/usr/src/linux# mkdir -p /etc/splash
filth:/usr/src/linux# cp /home/user/Debian-Mist_rev1.tar.bz2 /etc/splash
filth:/usr/src/linux# cd /etc/splash
filth:/etc/splash# tar -xjvf Debian-Mist_rev1
filth:/usr/src/linux# splash_geninitramfs -g usr/initramfs_data.cpio.gz -r 1024x768 Debian-Mist
```

Il tema che ho utilizzato io è Debian-Mist, davvero molto carino! Se non riceverete nessun messaggio

dalla shell, allora tutto è andato liscio. Adesso compiliamo l'immagine del kernel e i relativi moduli, con questi comandi:

```
filth:/usr/src/linux# make -j4 bzImage
filth:/usr/src/linux# make modules
filth:/usr/src/linux# make modules_install
```

Aspettate che tutto finisca e preparatevi alla configurazio del boot loader, nel mio caso, LiLo.

6)Configurazione LiLo

Adesso passiamo alla configurazione del bootloader. Diciamo che è la parte meno noiosa, e laboriosa di tutta sta gran operazione. Come prima cosa, copiate nella cartella /boot la configurazione appena generata dal kernel, il relativo System.map ed infine, l'immagine del kernel appena compilata:

```
filth:/usr/src/linux# cp System.map /boot/System.map-2.6.10nitro2
filth:/usr/src/linux# cp .config /boot/config-2.6.10nitro2
filth:/usr/src/linux# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.10nitro2
```

Adesso, con il vostro editor di testo preferito, editate il file di configurazione di lilo che trovate in /etc/lilo.conf , aggiungendo le seguenti linee:

```
image=/boot/vmlinuz-2.6.10nitro2
        label=Sid2.6.10
        append="video=vesafb:ypan,1024x768-32@60 splash=verbose,theme:Debian-Mist"
        read-only
#       restricted
#       alias=1
```

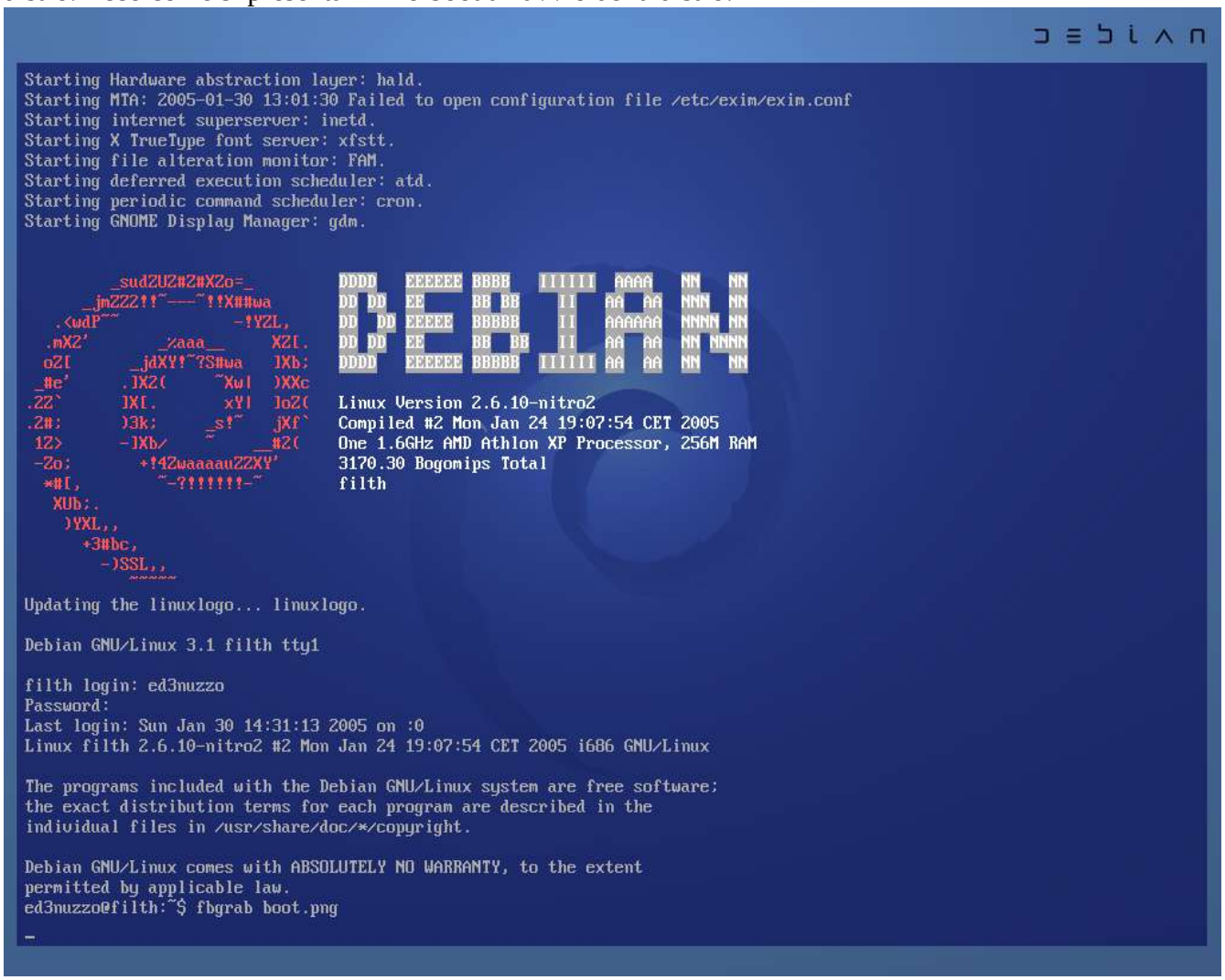
Così facendo, avrete configurato il boot loader per caricare l'immagine del nuovo kernel, con il relativo append per il caricamento del tema di boot. Altra cosa importante è quella di settare alla linea vga= il valore “normal”, altrimenti il tema non potrebbe partire. Se avete modificato tutto il necessario, reinstallate il bootloader con il comando:

```
filth:/usr/src/linux# lilo -v
```

Infine, riavviate :)

7)Conclusione

Dopo questi “IMMANI” sforzi, se tutto è andato bene, avrete il boot della vostra distro un po' più allegro del solito, e non più unicamentre strapieno di scritte di configurazione e script di init della distro. Ecco come si presenta il mio boot all'avvio della distro:



Guida pubblicata su http://linux.le.it
project by http://russo.le.it
powered by Vito Russo