Debianizzati

e-zine

GNU/Linux

GNU/Hurd

prodotta da www.debianizzati.org

numero 0 - Aprile 2009

Indice

Capitolo 1

Editoriale

Debianizzati.org è una comunità che si prefigge lo scopo di diventare un punto di riferimento per gli utenti Debian italiani [1]; rispetta un contratto sociale [2] in linea con il contratto sociale di Debian GNU/Linux [3]. Il progetto e-zine incominciò più di un anno fa...

> *Correva l'anno 2008, febbraio. Da una discussione animata della comunità il desiderio di scrivere una rivista sul nostro sistema operativo preferito si faceva sempre più intenso. L'idea nacque quasi per caso, "sputata" da un momento all'altro come i discorsi che nascono in un bar - "e se facessimo una rivista italiana su debian?" - Senza neanche finire la frase, già si accumulavano i topic con bozze e pensieri che andavano ad alimentare la voglia e la motivazione di creare un progetto... e così fu.*
>
> *Una nuova sezione del forum venne creata per dar vita al progetto. L'inizio fu rapido e ogni utente che raggiungeva il gruppo portava nuove idee ed energia che sommandosi fra loro, incominciavano a stabilizzare le fondamenta. Un sistema preciso d'archiviazione diede l'avvio ufficiale alla "produzione" degli articoli. Dopo un paio di mesi, questa si interuppe però in modo brusco... Il progetto "e-zine", poco tempo dopo la sua nascita, subì un forte rallentamento. Come ogni buona idea, l'inizio fu facile, la continuità minacciava però di non farsi vedere. A soli pochi mesi dopo il primo pensiero, l'ultimo stava già bussando alla porta... quando pochi giorno dopo, entrò addormentando il progetto.*

[1] cos'è debianizzati.org : http://guide.debianizzati.org/index.php/Cos'è_Debianizzati.Org
[2] contratto sociale di debianizzati.org : http://guide.debianizzati.org/index.php/Contratto_sociale
[3] contratto sociale debian : http://www.debian.org/social_contract.it.html

Ci volle quasi un anno per riprendere i lavori. Il team che lavorava al progetto venne rinforzato e riorganizzato con nuove entrate. Gli articoli riempivano rapidamente i nuovi capitoli, la struttura veniva definita man mano che una linea si designava a mostrare la via che avremmo percorso. Al momento di definire il telaio della rivista arrivò l'idea che caratterizza il progetto "e-zine": al posto di proporre un layout "statico", come una qualsiasi rivista cartacea, venne ideata una struttura particolare: prendeva la praticità di consultazione da una pagina web e la bellezza grafica da una rivista tradizionale: nacque lo "web-zine".

Per rendere il progetto il più "compatibile" possibile, l'e-zine si suddivide nello *web-zine* (versione online e versione scaricabile per la consultazione offline) e in una *versione stampabile* (pdf).

Colgo infine l'occasione per ringraziare tutti i ragazzi del progetto per l'ottimo lavoro svolto e speriamo di deliziarvi con le pagine a seguire... e i prossimi numeri dell'e-zine. Buona lettura!

brunitika, coordinatore e-zine

Capitolo 2

Storia e filosofia di Debian

In questa sezione verranno proposti articoli a riguardo la storia e la filosofia che sta dietro al sistema operativo debian.

Storia e filosofia di debian vuole approfondire le basi storico/culturali informatiche, legate alla distribuzione GNU/Linux forse più libera che c'è.

A seguire, per il numero inaugurale della rivista, un articolo sulla nascita di debian GNU/Linux.

2.1 La nascita di Debian

Era una calda serata d'estate. Nella città di West Lafayette, nella contea di Tippecanoe, nel nord-ovest dell'Indiana, Ian, Ian Murdock, studente d'informatica alla Purdue University, ammirava la sua bella Deb, Debra, mentre si riposava sul terrazzo della palazzina all'interno del campus. La luce del sole del tramonto si rifletteva sulla sua pelle ambrata e ne accentuava i lineamenti. Ian era soddisfatto. Dopo mesi di lavoro aveva appena finito di scrivere le ultime righe del suo annuncio al newsgroup *comp.os.linux.development*; il suo nuovo sistema operativo basato sul kernel Linux e il progetto GNU era arrivato a capolinea, dando vita alla prima versione: la 0.0.1. Dedicato a se stesso e alla sua fidanzata, lo nominò debian, appunto da Deb e Ian. Era il 16 agosto del 1993.

> *This is just to announce the imminent completion of a brand-new Linux release, which I'm calling the Debian Linux Release. This is a release that I have put together basically from scratch; in other words, I didn't simply make some changes to SLS and call it a new release. I was inspired to put together this release after running SLS and generally being dissatisfied with much of it, and after much altering of SLS I decided that it would be easier to start from scratch. The base system is now virtually complete (though I'm still looking around to make sure that I grabbed the most recent sources for everything), and I'd like to get some feedback before I add the fancy stuff.*[1]

Qualche anno prima, mentre Richard Stallman, creatore della free sofwtare foundation, lavorava assiduamente al progetto GNU[2] per sviluppare un sistema operativo completo sulla base del microkernel mach, nell'agosto del 1991 Linus Torvalds dava l'annuncio al comp.os.minix[3] che stava sviluppando un sistema operativo (ndr. per altro un kernel) ispirato a minix (creato nel 1987 da Andrew S. Tanenbaum, professore di informatica all'università Vrije di Amsterdam, su struttura a mikrokernel [unix-like] a scopo didattico);

[1] annuncio ufficiale completo al newsgroup comp.os.linux.development

[2] il sistema operativo libero al quale mirava la Free Software Foundation di Richard Stallman

[3] annuncio ufficiale al newsgroup newsgroup comp.os.minix di Linus Torvlads, sull'imminente sviluppo di quel che sarà Linux.

si trattava dell'ora conosciutissimo Linux [4] . Il porting di *bash* e *gcc* già nella primissima versione testimoniavano una collaborazione con il progetto GNU agli albori del primo sistema operativo alternativo unix-like completo. Sì, proprio perchè il "sistema" Linux non era nient'altro che un kernel, mentre il "sistema" GNU era un sistema operativo ma senza kernel. Dal momento che il progetto Hurd, basato sul microkernel mach, stentava a decollare, la free software foundation di Stallman decise di adottare il Kernel Linux come Kernel del sistema operativo GNU: nacque GNU/Linux. Questo termine venne però coniato solo con la nascita di debian...

Nel 1992, Peter MacDonald creò forse la prima distribuzione GNU/Linux: Softlanding Linux System (SLS)[5]. Oltre al kernel Linux e le utilità di base, la SLS comprendeva una collezione di softwares per la gestione del sistema, tra i quali un sistema grafico e il protocollo *TCP/IP*. Grazie alla sua efficenza, la SLS dominò il mercato in quegli anni diventando la distribuzione più popolare. Agli inizi del 1993 ci furono i primi scontenti, in particolare dopo che gli sviluppatori decisero di cambiare il formato eseguibile da *a.out* a *ELF*. Fu così che un certo Patrick Volkerding, allora studente presso l'università di stato del Minnesota, dopo aver apportato alcune modifiche alla SLS decise di creare una nuova distribuzione basata su quest'ultima. Il 16 luglio del 1993, Volkerding annunciava al newsgroup comp.os.linux la nascita del suo lavoro: Slackware 1.00. In quello stesso periodo, anch'egli dopo qualche modifica all SLS, Ian Murdock, anch'egli scontento, lavorava ad una nuova distribuzione; ma questa volta, partendo da zero.

A Ian Linux piacque tantissimo e si fece coinvolgere da subito. Più che altro, gli piaceva la comunità che gli stava attorno e decise di dare il suo contributo a questo progetto. La motivazione e lo slancio iniziale furono però in parte placati da una serie di distribuzioni mal realizzate (allora non si parlava di distribuzioni nel significato attuale, quanto a "revisioni" del kernel Linux con utilità di base). Ian decise allora che l'unica soluzione era di fare una cernita del software libero al momento disponibile, per creare una distribuzione personale, libera da additivi. Cercando l'interesse della comunità scrivendo nei newsgroup, Ian scoprì di avere fra i primi interessati un "ospite d'onore": Richard M. Stallman[6]. La Free Software Foundation (FSF), che fino a quel punto aveva ignorato Linux, si interessava agli

[4] Si consiglia la lettura di "Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi)", *Linus Torvalds e David Diamond*, edizione Garzanti

[5] Questa distribuzione non esiste più. Per un po' di "storia", un vecchio readme della versione 1.03.

[6] Richard M. Stallman è da considerarsi come il creatore del movimento del sofwtare libero. Il link vi rimanda alla sua pagina personale. Qui, un articolo su wikipedia con tutte le informazioni sul personaggio.

sviluppi di Linux ed in particolare al progetto di Murdock, che veniva dunque considerato in linea con la propria filosofia. Fu proprio questo slancio dato dalla FSF a far decollare debian e suscitare l'interesse di tutta la comunità; era l'inizio del progetto debian, come lo chiamò lo stesso Murdock, "Debian Linux Release". La stessa società finanzierà poi il progetto dal 1994 al 1995.

Il 27 agosto del 1993, Ian annunciò al newsgroup *comp.os.linux.development* le sue idee su come intendeva organizzare il progetto debian. Da come era stato affascinato dalla comunità nascente di Linux, dinamica ed efficente, ma soprattutto aperta, decise di utilizzare lo stesso modello, lasciando la possibilità a qualsiasi sviluppatore di partecipare al progetto. Lui stesso, si sarebbe definito coordinatore; chiunque avrebbe poi potuto contribuire.

> *I would like to point out here that I would like this distribution to develop in the same way as much of the rest of Linux has developed. In other words, I want everyone to *contribute* to this effort and not simply use something that one man or team has put together. This distribution will be improved by the Linux community as a whole, and I will simply serve as the coordinator of the effort.*[7]

Fra l'agosto e il dicembre del 1993 vennero rilasciate le versioni dalla 0.0.1 alla 0.90. Nel gennaio del 1994, in contemporanea con l'uscita della versione 0.91, Ian scrisse il "Manifesto debian" sulla base dello spirito del "Manifesto GNU" di Stallman. Il documento riassumeva il significato e la filosofia di debian, allora "Debian Linux Release". Il manifesto fu diviso in tre sezioni: cosa è debian Linux, perchè debian è stata realizzata, come debian tenta di risolvere queste problematiche. Tra i punti più importanti risalta l'apertura dello sviluppo a tutta la comunità informatica e la collaborazione con la Free Software Foundation, il buon auspicio di creare una distribuzione facile da utilizzare e ben mantenuta senza mai diventare un prodotto commerciale, il continuo sviluppo del sistema con la possibilità di competere sul libero mercato.

Una lettura consigliata è sicuramente anche "codice libero", scritto da Sam Williams, tradotta da Bernardo Parrella, disponibile liberamente in rete o in formato cartaceo, edizione Apogeo.

[7] annuncio sull'organizzazione di debian completo al newsgroup comp.os.linux.development

Poco dopo l'uscita del manifesto debian, la Free Software Foundation avanzò la prima richiesta: voleva che Murdock chiamasse la propria distribuzione GNU/Linux (dopo che "Lignux", la prima proposta, era stata talmente contestata dalla comunità Hacker fino al punto da convincere Stallman a cambiare nome). Ian, già devoto alla causa del software libero, accettò la proposta, soprattutto per calmare le acque fra Linux e GNU. Sì, anche a non crederlo, in quel periodo si stava profilando una diatriba fra la comunità di sviluppatori del kernel Linux e quella del progetto GNU. Eppure i due sistemi erano complementari... Grande scompiglio fu creato dalla librerie *glibc* (GNU C Library), con le quali è possibile effettuare delle "chiamate di sistema" direttamente al kernel. Data la grande domanda di nuove funzioni che venivano implementate nel kernel da parte degli sviluppatori, la GNU non riusciva quasi a tenere il passo con le glibc. Alcuni sviluppatori di Linux proposero di creare un fork delle *glibc* per il kernel Linux. Le due comunità potevano dividersi e ancora peggio diventare un giorno incompatibili. Per fortuna l'unità venne mantenuta e il nome GNU/Linux venne coniato per il nuovo sistema operativo. Se sia stato Murdock ad evitare la scissione fra GNU e Linux accettando la proposta di Stallman nessuno lo può dire; in ogni caso, Ian Murdock ebbe un ruolo importante in questi primi anni di Linux e della "fusione" con la GNU.

Era una fredda mattina d'inverno dell'anno 1994. Ian si svegliò turbato; era stato tutto un sogno? Aprì gli occhi: Debra stava adagiata al suo fianco. Il suo respiro era lento e regolare, dormiva. In quel momento si ricordò la decisione presa il giorno prima; il suo progetto era lanciato, finanziato dalla Free Software Foundation e decine di sviluppatori già contribuivano al suo sviluppo. L'idea era buona, non restava che andare avanti.

E fu così che Debian GNU/Linux prese piede nella storia dell'informatica. Negli anni a venire seguirono focose discussioni con Stallman e ulteriori sviluppi del sistema... ma questa storia ve la racconterò un'altra volta.

2.2 Coda e conclusioni

La parte iniziale e finale del racconto, ad eccezione delle date e dei nomi, sono fantasiose per dare un carattere "romanzesco" alla narrazione (e non annoiarvi con una semplice biografia di Ian Murdock). Tutti gli altri fatti, così come i nomi e le distribuzioni citate, si basano su dati reali. Non ci sarà difficile allora capire come Ian Murdock e la sua debian, siano proprio alla base della storia del software libero "moderno"; se daremo un'occhiata alle distribuzioni più conosciute/scaricate al momento, non ci sorprenderemo di vedere ubuntu al primo posto in classifica... e sapendo tutti su cosa si basa quest'ultima distribuzione, ci renderemo conto di come la nascita di debian, dia ancora oggi una grande spinta verso l'espansione e la conoscenza di GNU/Linux, così come all'apertura al software libero.

2.3 Bibliografia

Oltre a quanto già citato a margine del racconto:

siti Internet

http://www.gnu.org/gnu/linux-and-gnu.it.html
http://www.debian.org/doc/manuals/project-history/index.it.html
http://blog.thedebianuser.org/?p=153

newsgroup

comp.os.minix
comp.os.linux
comp.os.linux.development

libri

- *Sam Williams, Codice libero (Free as in freedom). Richard Stallman e la crociata per il software libero, Apogeo.*
- *Linus Torvalds e David Diamond, Rivoluzionario per caso. Come ho creato Linux (solo per divertirmi), Garzanti.*

Capitolo 3

Il sistema operativo Debian

In questa sezione, tutto sul sistema operativo debian GNU/Linux.

Il 14 febbraio 2009 viene rilasciata la nuova versione stable di debian.
La nuova release si chiama lenny ed è la versione 5.0.
Nel prossimo articolo scopriremo come installarla sul nostro computer sfruttando il metodo grafico.

3.1 Installazione grafica di Debian Lenny 5.0

Questo articolo vuole affrontare il processo di installazione di Debian GNU/Linux, in particolare Debian Lenny: l'attuale versione stable.
La guida è pensata per tutti gli utenti che si vogliono avvicinare a Linux e/o a Debian GNU/Linux (da adesso in avanti, più semplicemente, Debian) e cerca di spiegare, passo per passo, tutto il necessario per giungere alla fine dell'installazione del sistema operativo e di dare una infarinatura sui termini e le nozioni più diffuse e più utili per affrontare al meglio il nuovo sistema operativo.

3.1.1 Ottenere Debian

La versione stabile di Debian è ottenibile tramite il sito ufficiale.
Sono presenti diversi supporti, vediamo di scegliere quello a noi più utile:

- **Netinstall** : si tratta di una versione del debian-installer (così si chiama il sistema di installazione di Debian) minimale, occupa 180Mb, che contiene i pacchetti necessari ad una installazione minimale... per tutto il resto è necessaria una connessione ad internet. È consigliata per tutti gli utenti che dispongono di una connessione Adsl con router Ethernet (caso in cui non è necessario effettuare nulla per connettersi ad internet).

- **Business Card** : questa versione è pensata per essere messa su i cd Business Card ed ha una dimensione di circa 40Mb. A differenza della NetInstall scarica da internet anche i componenti del debian-installer.

- **CD/DVD tramite Http/ftp/BitTorrent/Jigdo** : se non si ha una connessione a banda larga a casa è possibile scaricare le iso tramite uno dei protocolli indicati:

  - http/ftp: i protocolli più usati
  - Bittorrent: il famoso protocollo di download
  - Jigdo: un sistema molto comodo che approfondiremo prossimamente.

Da notare che non è necessario scaricare tutti i CD/DVD: per effettuare una installazione base bastano i primi 3 CD o il primo DVD.

Acquistare On-Line

Se non si dispone di una connessione a banda larga e nemmeno un amico ce la può prestare, è possibile acquistare i dvd/cd da uno dei negozi indicati qui:
http://www.debian.org/CD/vendors/#it

Scelta dell'architettura

Dobbiamo scegliere l'architettura adeguata per il nostro processore, sulla quale ci dovremo informare prima di procedere al download dell'immagine del CD/DVD di installazione. C'è da dire che le architetture più comuni sono **i386** e **amd64**. Un sistema operativo i386 a 32 bit può essere installato tranquillamente sia avendo un processore x86 compatibile che x86-64 compatibile, dato che questi ultimi mantengono ancora il supporto alle istruzioni x86. Se il nostro scopo è quello di installare un sistema operativo a **32 bit** dovremo quindi utilizzare l'installer per architettura i386 con entrambi i tipi di processore.
Se invece possediamo un processore capace di calcoli a **64 bit**, sia che implementi la tecnologia EM64T (*Extended Memory 64 Technology*) di Intel, che l'estenzione AMD64 di AMD e vogliamo installare un sistema a 64 bit, dobbiamo rivolgere la nostra attenzione verso il supporto per amd64.

Il processore avrà l'estensione x86-64? Nel caso raro in cui non si riesca a trovare informazioni in rete sul nostro processore possiamo provare una live GNU/Linux e interrogare il nostro sistema. Analizziamo la sezione flags del file /proc/cpuinfo alla ricerca del flag **lm** (*long mode*) che se presente è indice del fatto che il nostro processore è x86-64 compatibile.
In alternativa interroghiamo il sistema con il comando:

```
$ getconf LONG_BIT
```

Se il sistema restituisce la risposta **64**, il processore è x86-64 compatibile, altrimenti restituisce la risposta **32** indice del fatto che è solo x86 compatibile.

Cosa ci serve

Oltre al supporto per l'installazione dobbiamo avere a disposizione le seguenti cose:

- Il manuale della nostra scheda madre
- Un elenco (anche generico) dell'hardware sul nostro computer

Sebbene queste cose non siano fondamentali, ci potranno essere di aiuto nel caso fosse necessaria una installazione particolare o di livello avanzato.

Boot da CD-Rom/DVD

Il software che ci guiderà nel processo di installazione è presente sul CD/DVD, per cui bisogna configurare il computer affinché effettui il boot dal lettore ottico.
Per modificare il device dal quale viene effettuato il boot bisogna entrare nel programma di configurazione del bios, premendo in genere F2 o Canc nella schermata di presentazione del bios (normalmente viene visualizzato il logo del costruttore o della scheda madre oppure viene effettuato il conteggio/test della ram), e impostare nella sezione Boot Device il lettore CD/DVD come primario, e l'hard disk come secondario. È consigliabile consultare, in caso di dubbio, il manuale della propria scheda madre poiché la procedura varia molto da una scheda madre all'altra.
Una volta salvata la configurazione si può riavviare il computer, inserire il CD/DVD e ci si troverà davanti alla schermata di boot del debian-installer

3.1.2 Installazione grafica

Le seguenti diapositive con relative spiegazioni, sono state realizzate utilizzando l'immagine netinstall con il debian-installerRC2 rilasciato il 31 gennaio 2008.
Per DE è inteso Desktop Environment ovvero l'insieme di programmi che ci permettono di organizzare, visualizzare, creare e gestire i nostri file. I DE più usati sono GNOME (il DE di default in Debian) e KDE, i quali offrono molte opzioni ma anche un poco pesanti avendo bisogno di molte risorse. Per PC con poche risorse o per coloro che amano un desktop minimale sono disponibili XFCE e LXDE. Tutti e quattro i DE sono disponibili con l'installer debian, a noi la scelta...

Inserito il cd/dvd nel lettore, iniziamo l'installazione... questa è la prima schermata che vedremo.... se diamo "invio" partirà l'installazione in modo classico.

Col tasto freccia giù selezioniamo Graphical install per installare il DE GNOME, se invece desideriamo un altro DE: KDE, XFCE LXDE spostiamoci in -Advanced option

Proseguiamo dando “invio”

Selezioniamo -Alternative Desktop Environment

Selezioniamo il DE che preferiamo e diamo “invio”

Come prima dobbiamo decidere come procedere, se nella maniera classica o nel modo grafico, notare che in alto abbiamo KDE (che abbiamo selezionato precedentemente)... se, invece, vogliamo tornare indietro e cambiare...

Selezioniamo di nuovo -Advanced options

Selezioniamo -Back e ritorneremo al boot iniziale

Una volta deciso quale DE installare, procediamo... ci si presenterà la scelta del linguaggio da utilizzare per l'installazione

Selezioniamo il linguaggio che vogliamo usare e proseguiamo

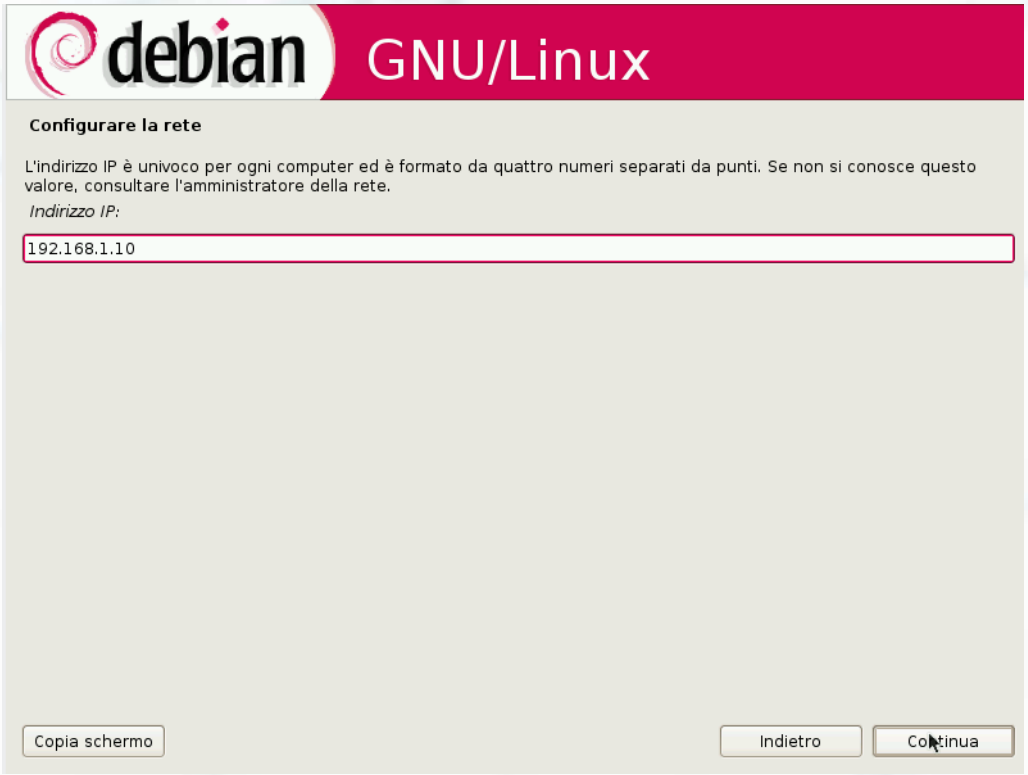
Selezioniamo la lingua o la regionalità

Ora scegliamo la mappa della tastiera che vogliamo utilizzare

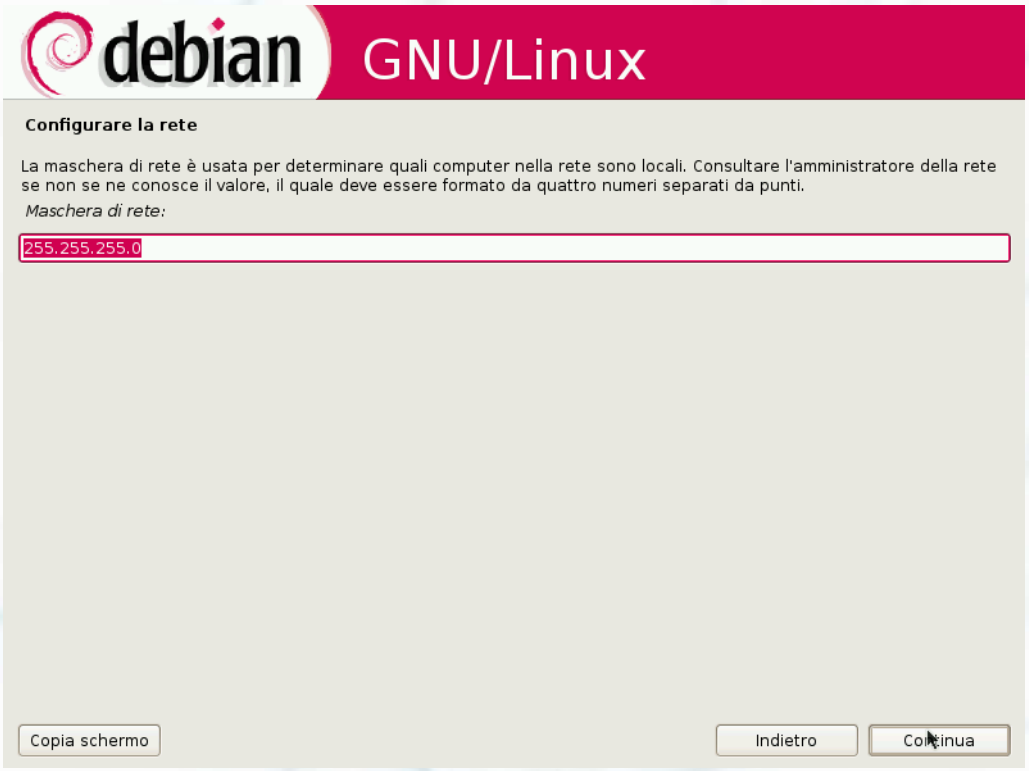
Subito dopo parte la configurazione automatica della rete con il DHCP, se la configurazione fallisce o vogliamo fare una configurazione manuale assegnando un ip statico e dns diversi dal gateway, cliccare su annulla e...

...selezioniamo - Configura la rete manualmente

Inseriamo il nostro ip statico



Automaticamente viene proposta la - Maschera di rete



...così come il gateway, ricordo che il gateway altro non è che l'indirizzo del router

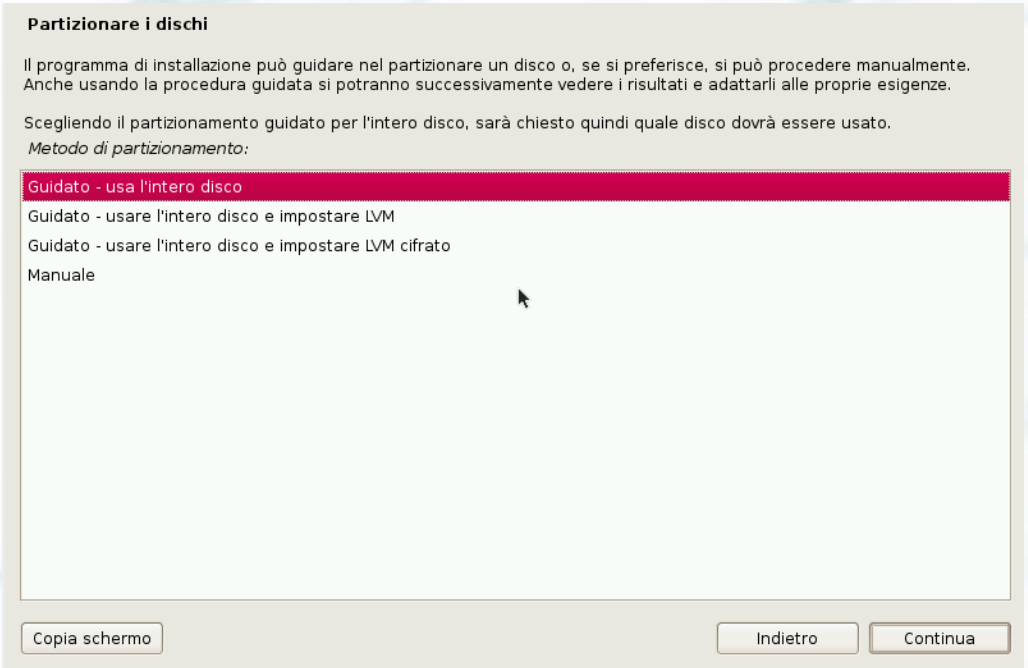
Come DNS ci viene proposto l'indirizzo del router, se si vuole metterne altri - tasto backspace e...

...inserire quelli del nostro provider o quelli che desideriamo...

Ora inseriamo il nome che vogliamo assegnare al nostro sistema

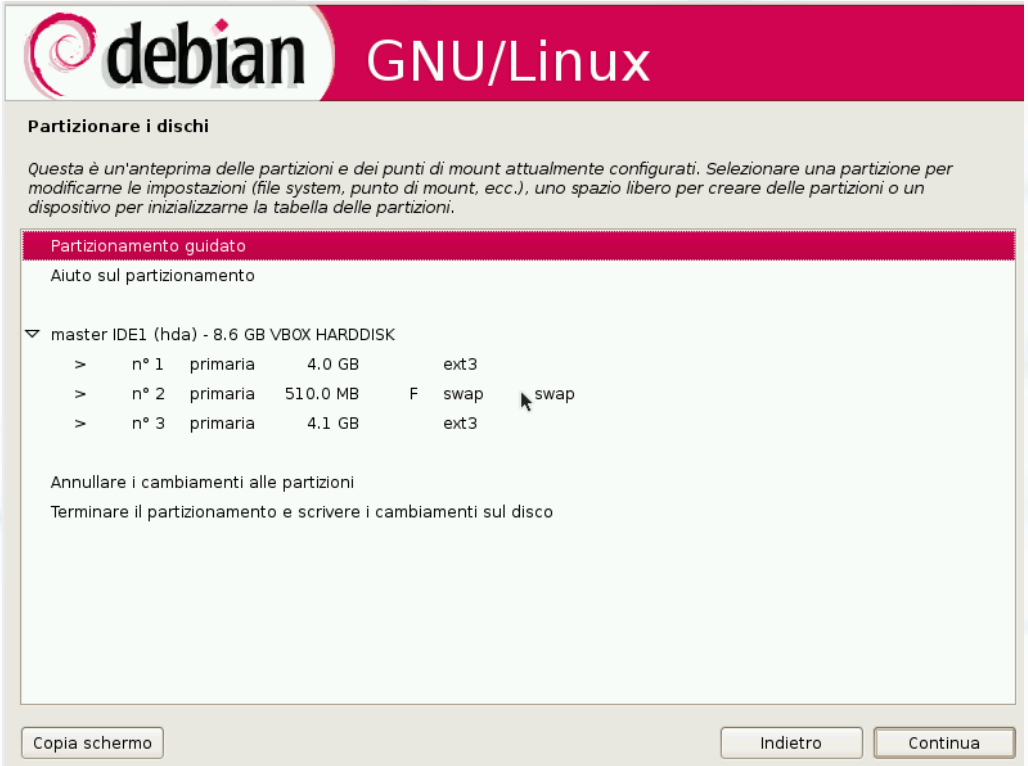
Aggiungiamo il nome del dominio, se ne abbiamo uno, oppure lasciare in bianco

Ora siamo nella fase più delicata, dobbiamo assegnare o creare la partizione che ospiterà la nostra Debian

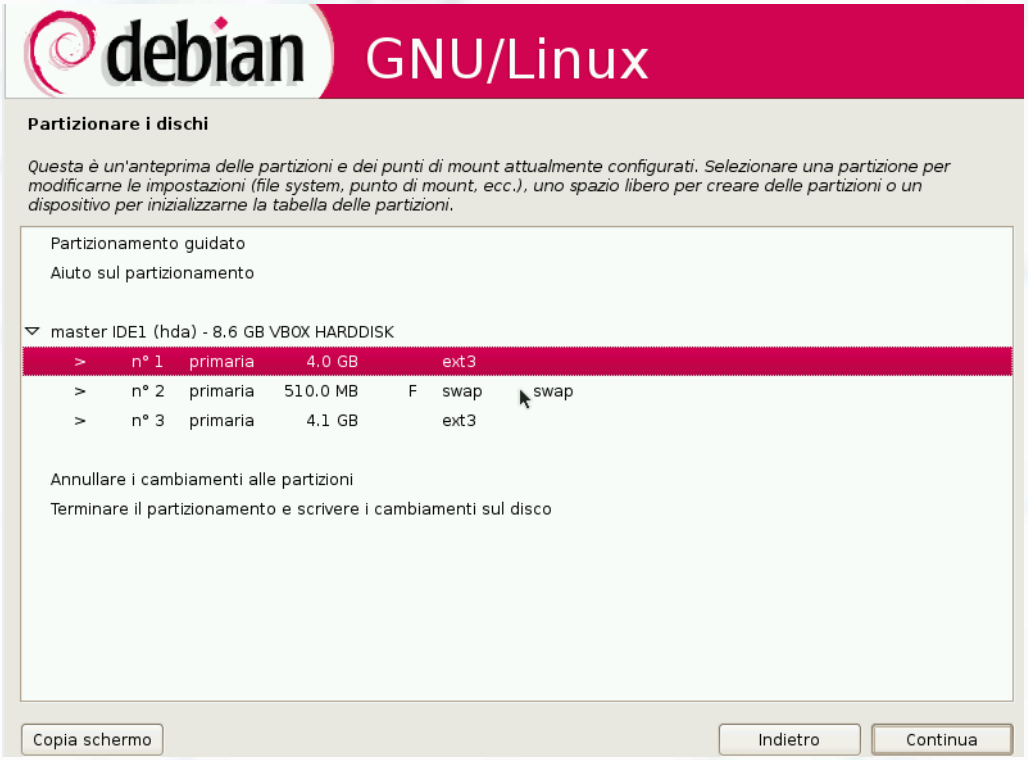


Se si ha a disposizione tutto un intero disco, possiamo scegliere la prima opzione, se invece abbiamo già installati altri SO o abbiamo una /home separata, e non vogliamo perdere nulla, selezioniamo manuale

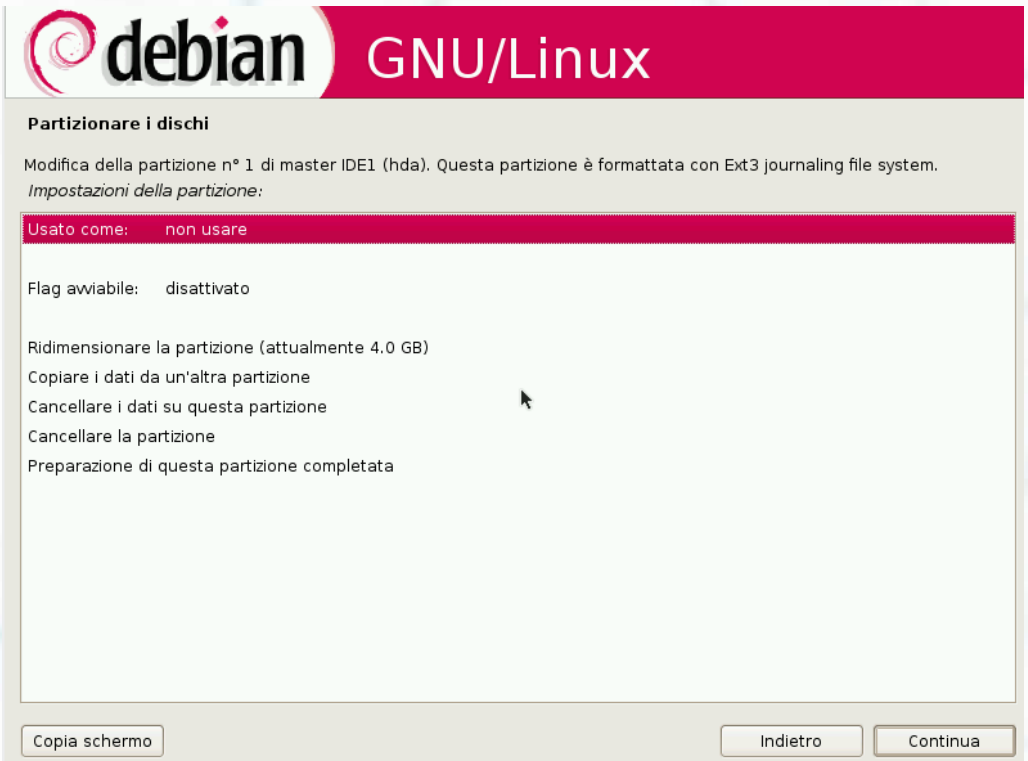
Come si vede ci sono tre partizioni già esistenti ma non sono assegnate



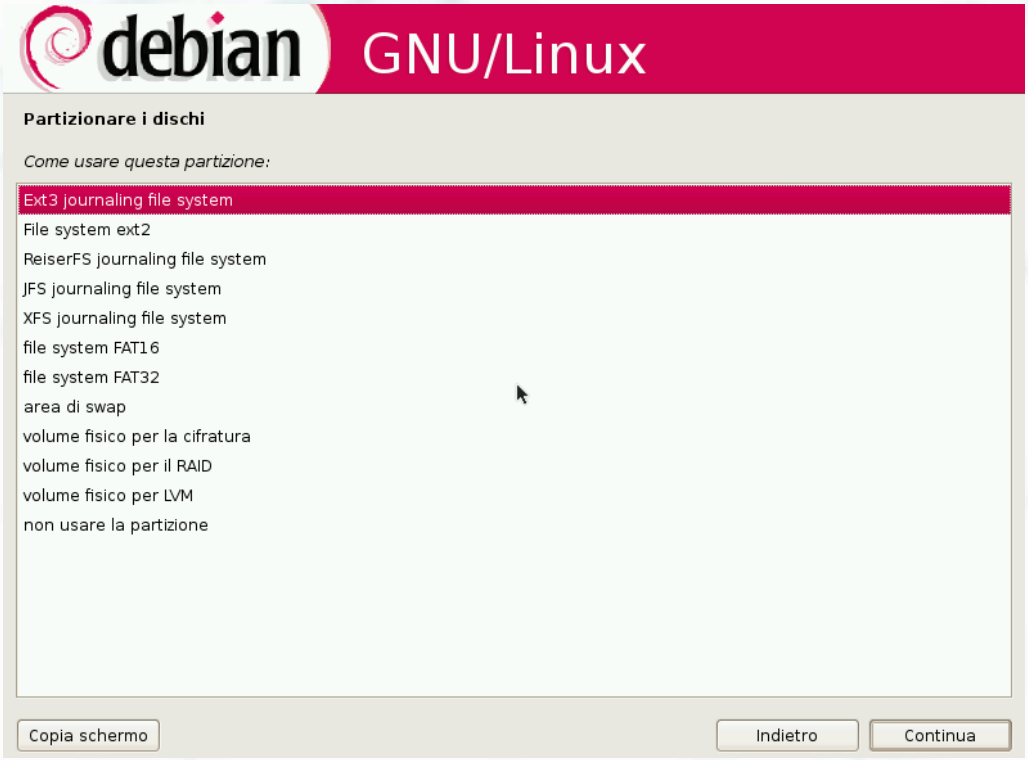
Selezioniamo la prima partizione e diamo “invio”



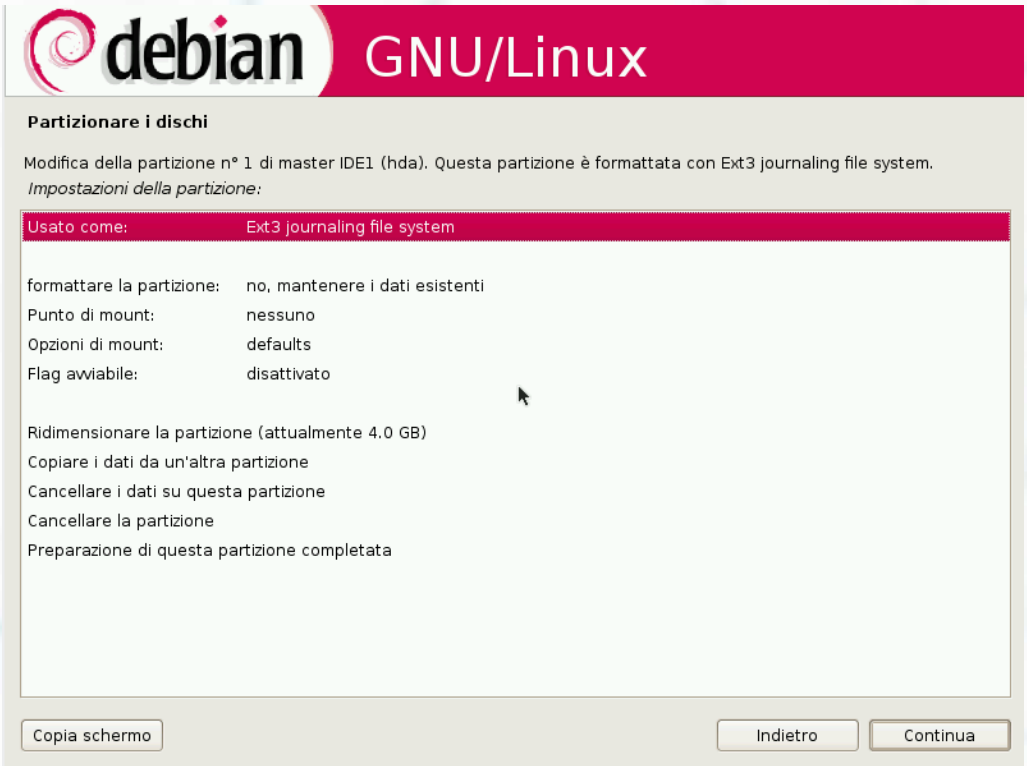
Evidenziamo come da immagine e “invio”



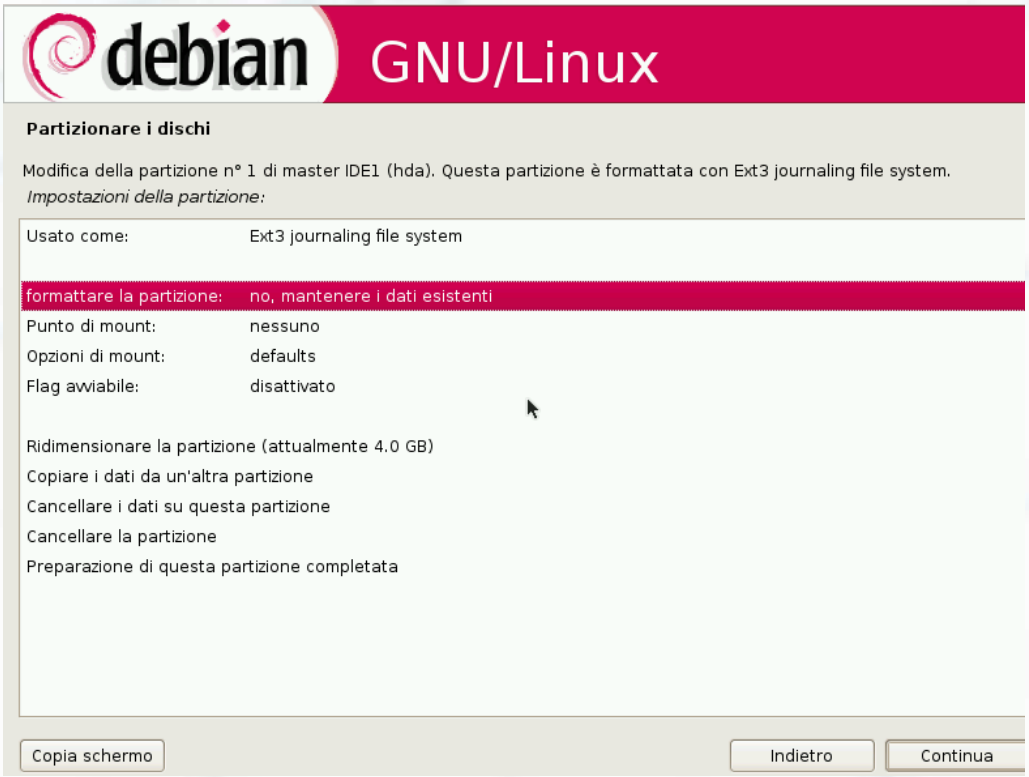
Selezioniamo il file system (ext3 è quello di default)



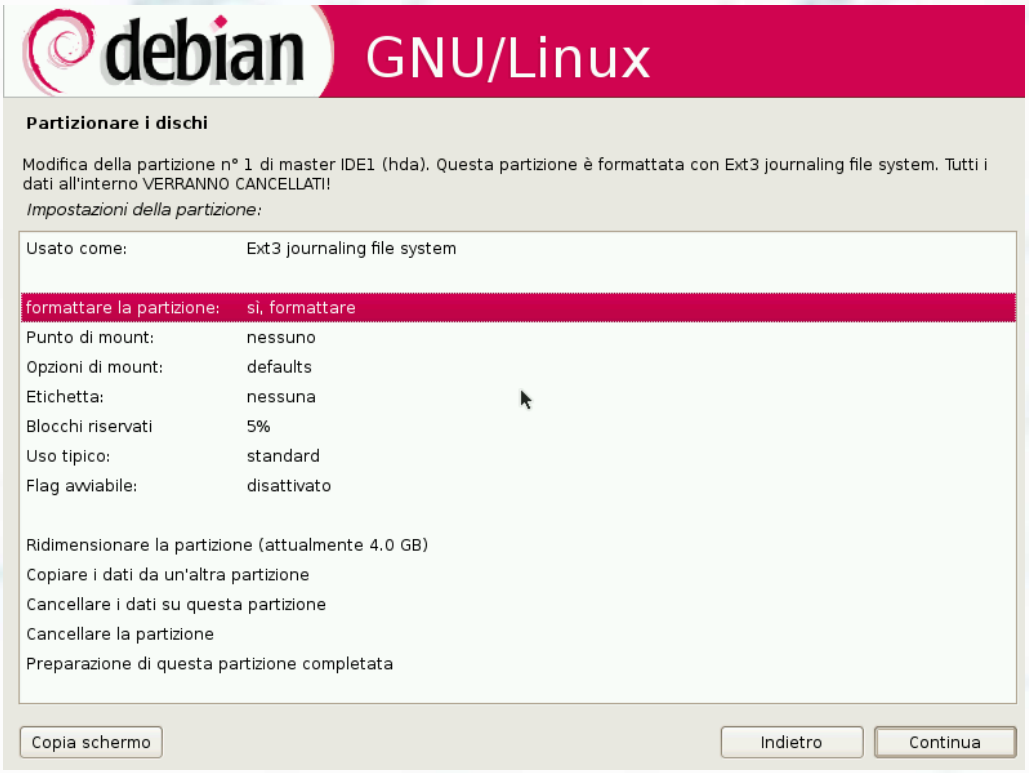
Vediamo evidenziato il file system che abbiamo scelto, ora spostiamoci alla seconda voce



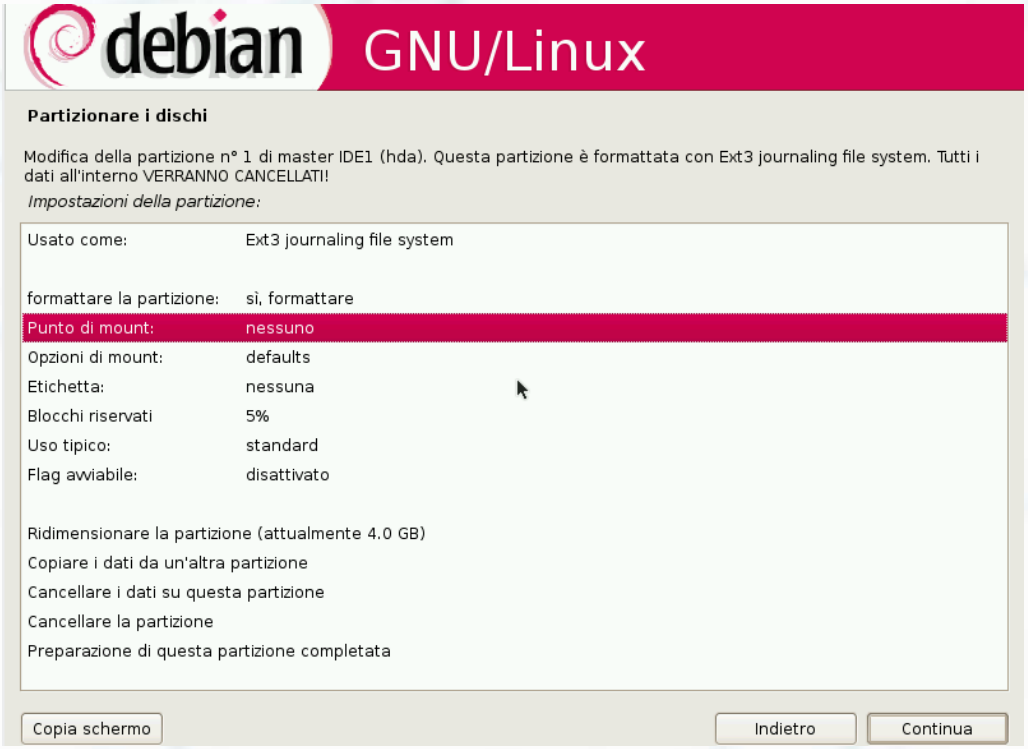
Se in questa partizione dobbiamo installare il sistema diamo “invio”



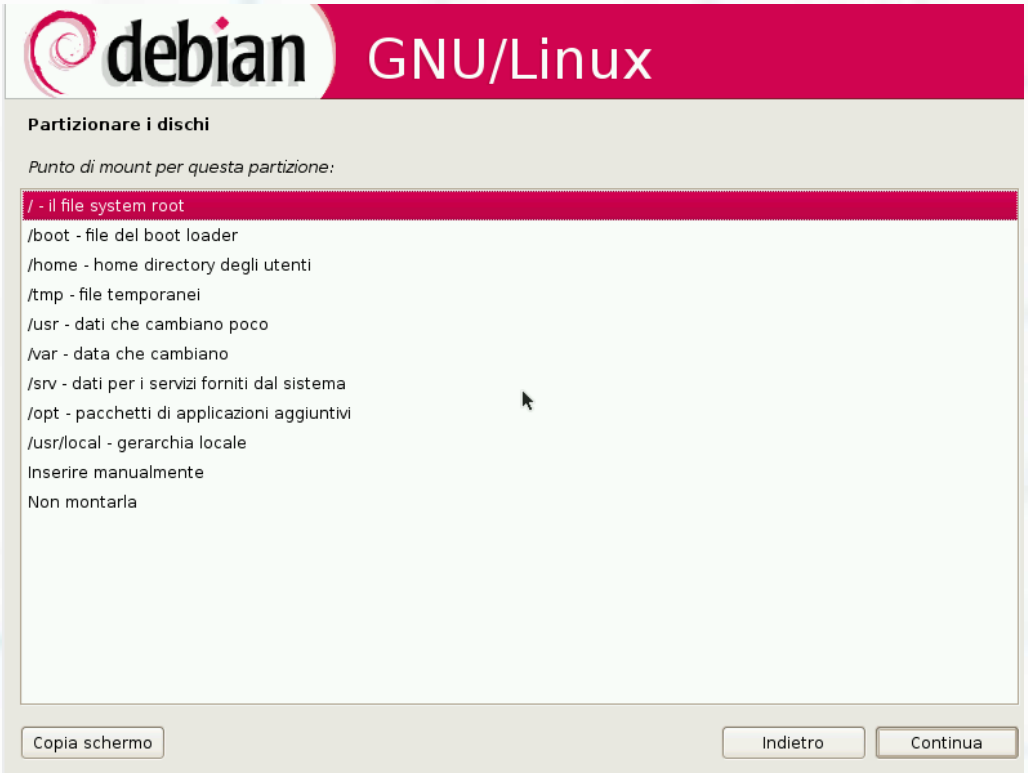
Vediamo che la partizione verrà formattata, **ATTENZIONE tutto il suo contenuto verrà cancellato!**



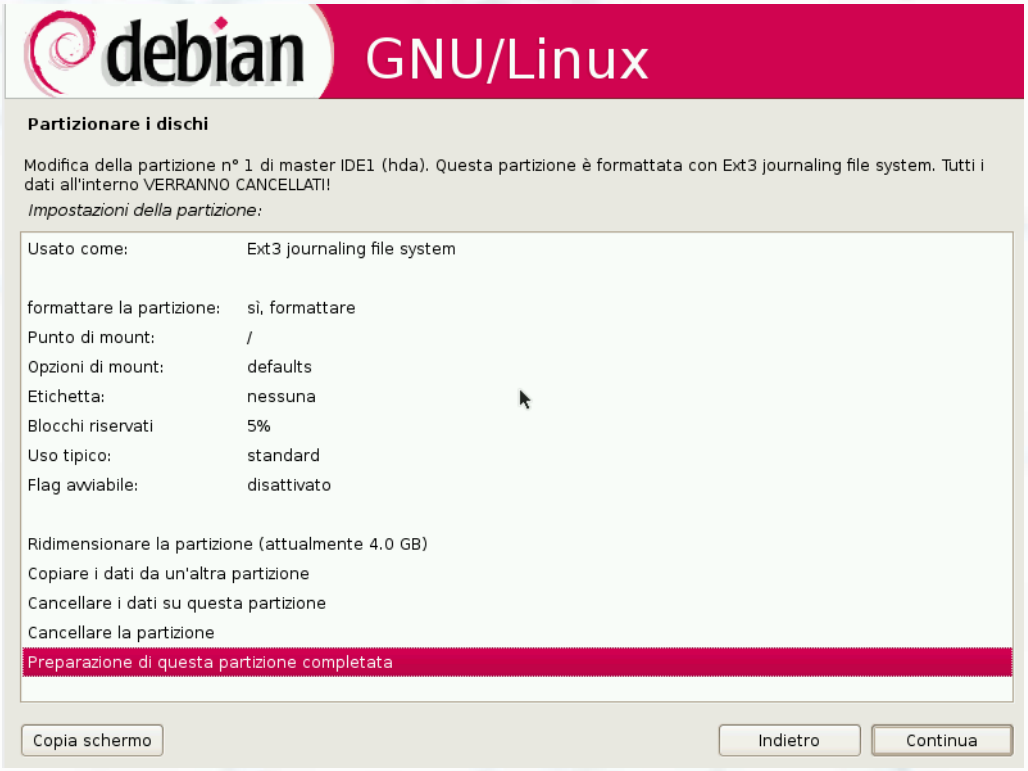
Ora dobbiamo scegliere il mount point



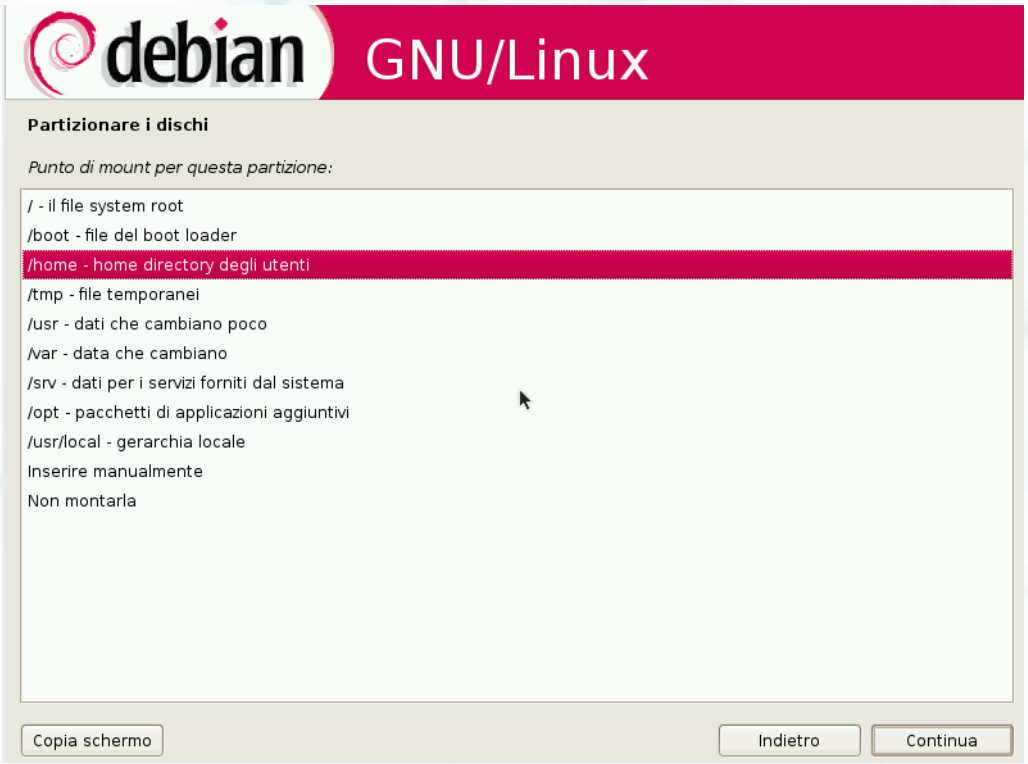
Scegliamo / root come da immagine



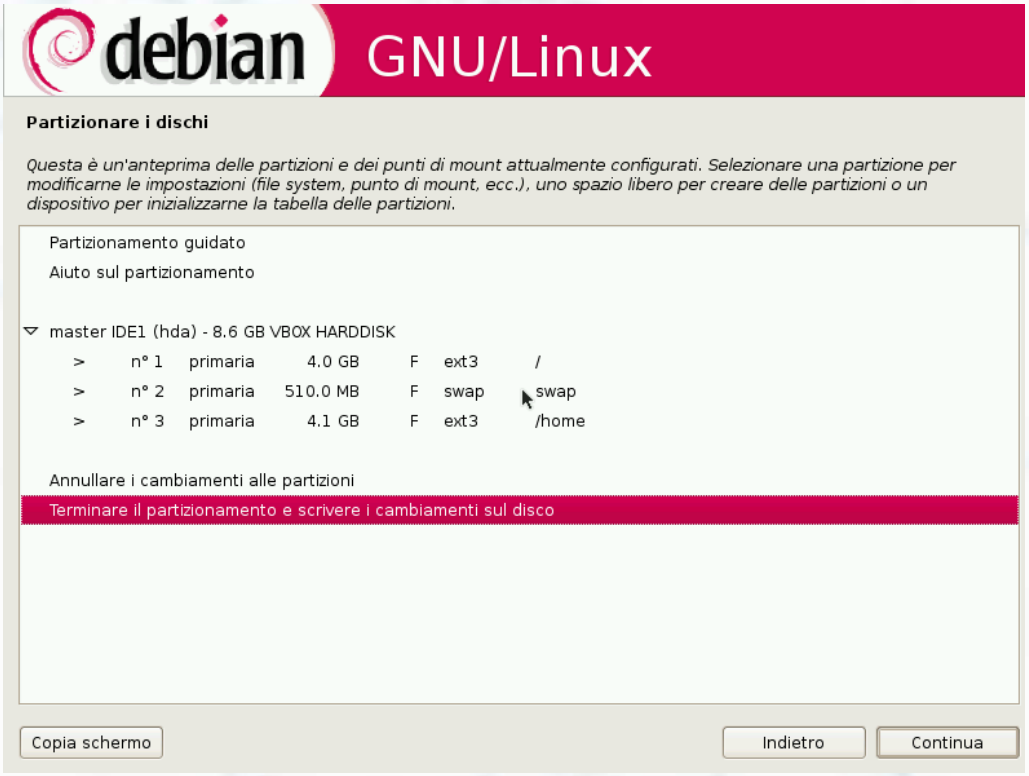
Eccoci al riepilogo della partizione che abbiamo creato, se va bene selezioniamo come da immagine e procediamo



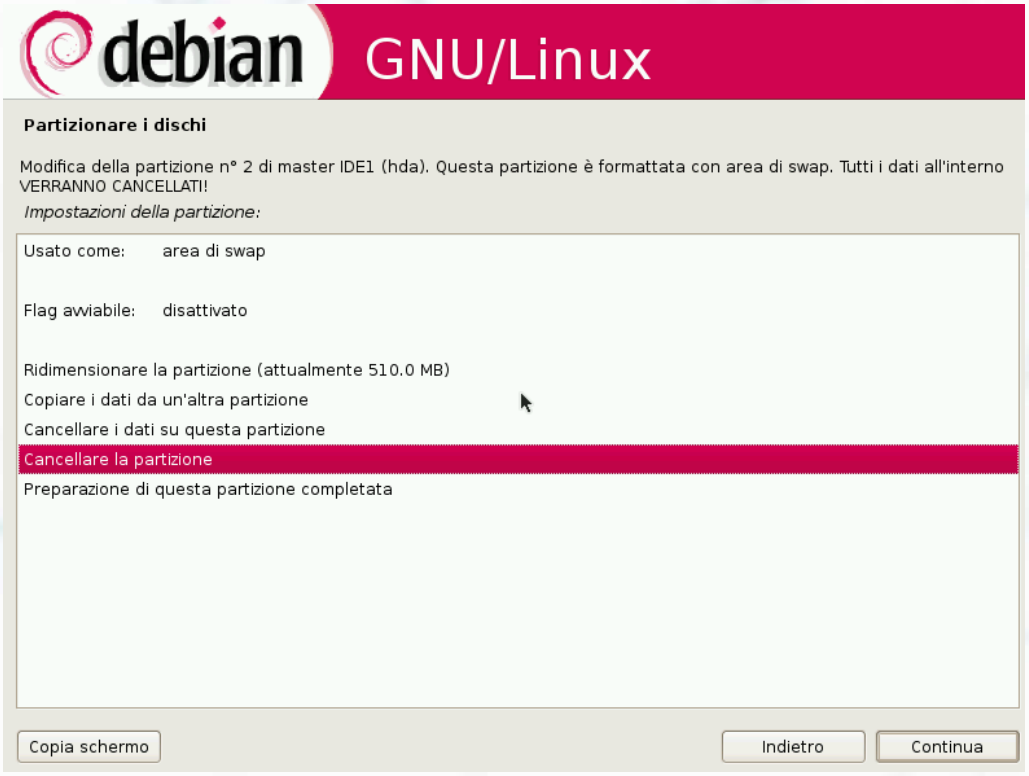
Rifacciamo lo stesso con le altre partizioni, avendo cura di selezionare i file system e i punti di mount da assegnare ad ogni partizione e andiamo avanti; se abbiamo già una /home e vogliamo mantenerla intatta - **non cambiare il file system e non formattare**



Alla fine abbiamo il riepilogo del nostro partizionamento, se tutto corrisponde ai nostri desideri andiamo avanti

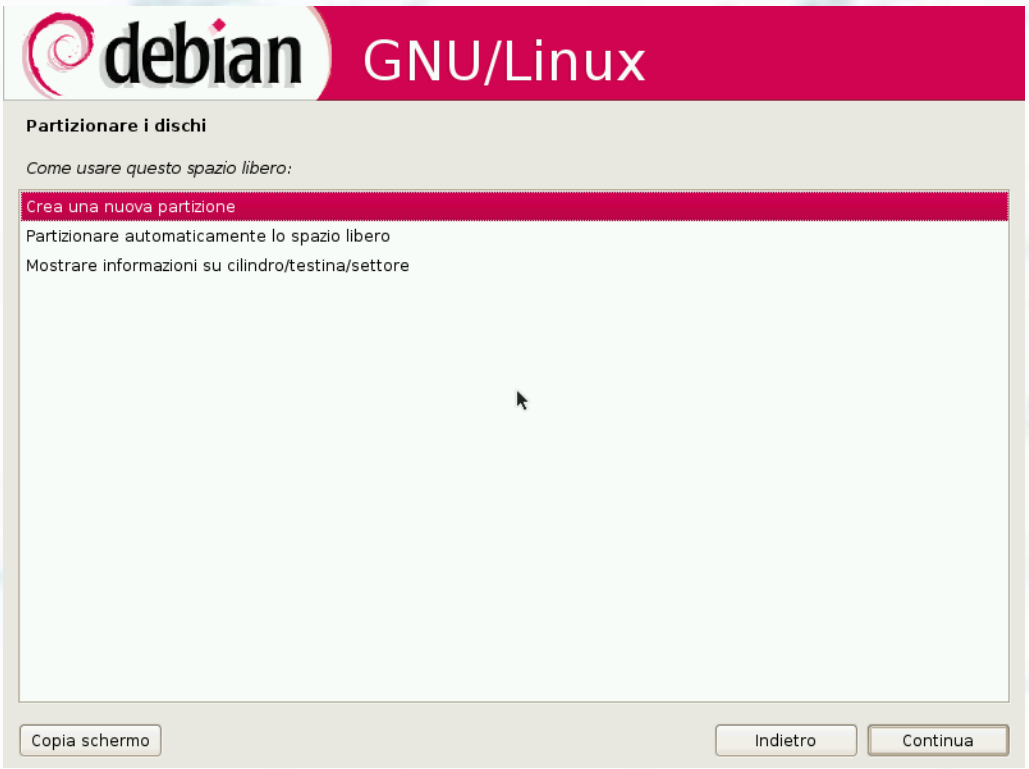


Se invece vogliamo cancellare tutte le partizioni e crearle ex-novo, selezioniamo una ad una le partizioni, ci portiamo su - Cancellare la partizione, come da immagine

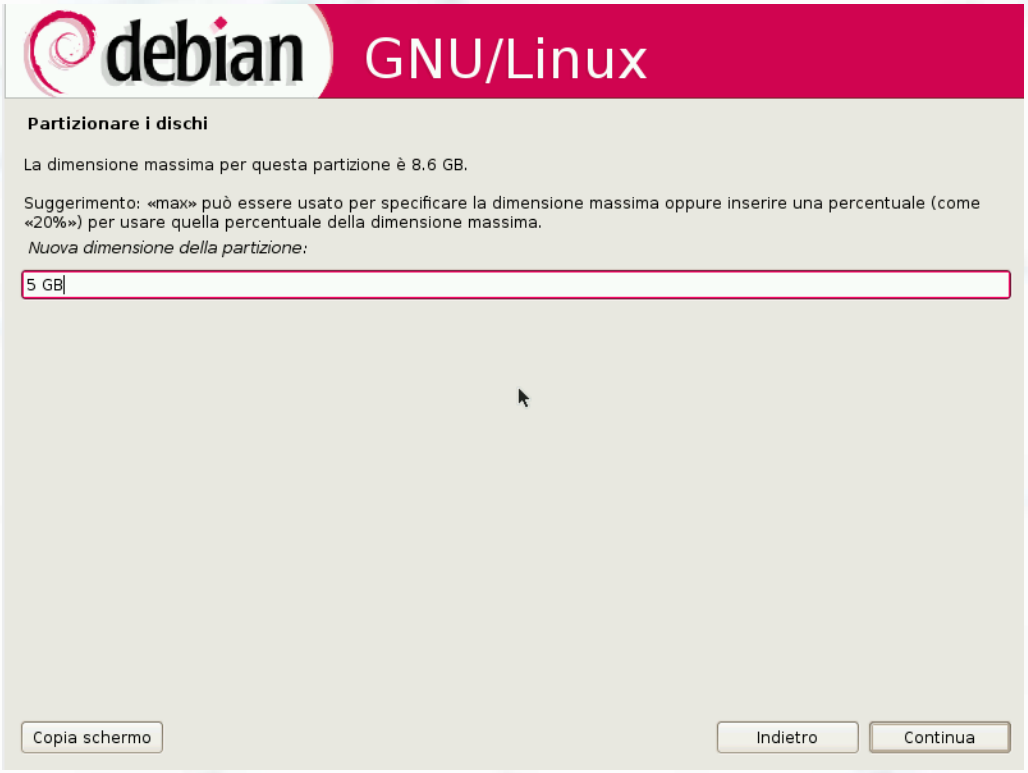


Alla fine avremo tutto lo spazio a disposizione, andiamo avanti per creare le nuove partizioni

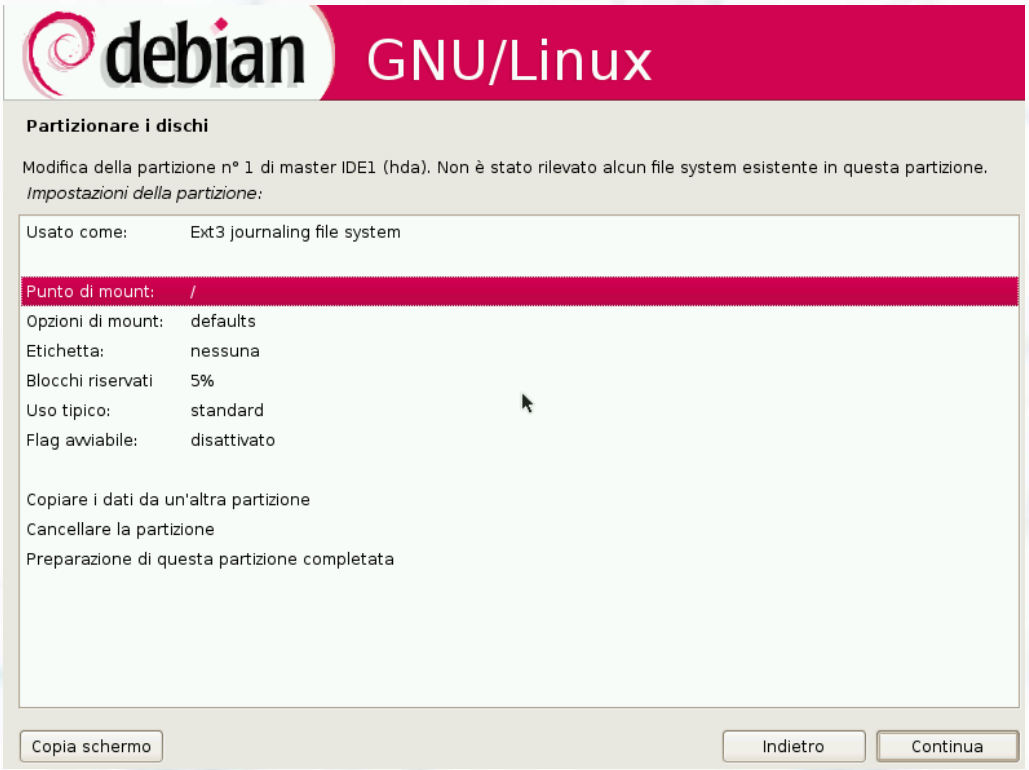
Seguono immagini su come proseguire



Scegliamo la dimensione, per /root (10-15 GB) se non si hanno problemi di spazio

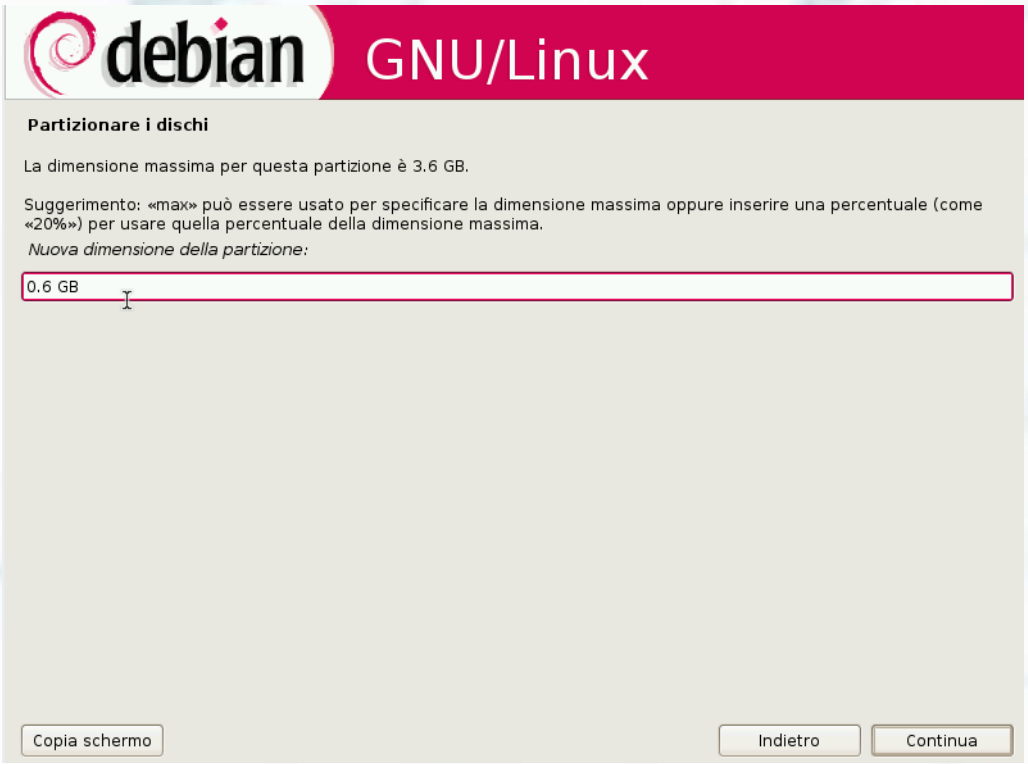


Viene creato il file system e il punto di mount automaticamente

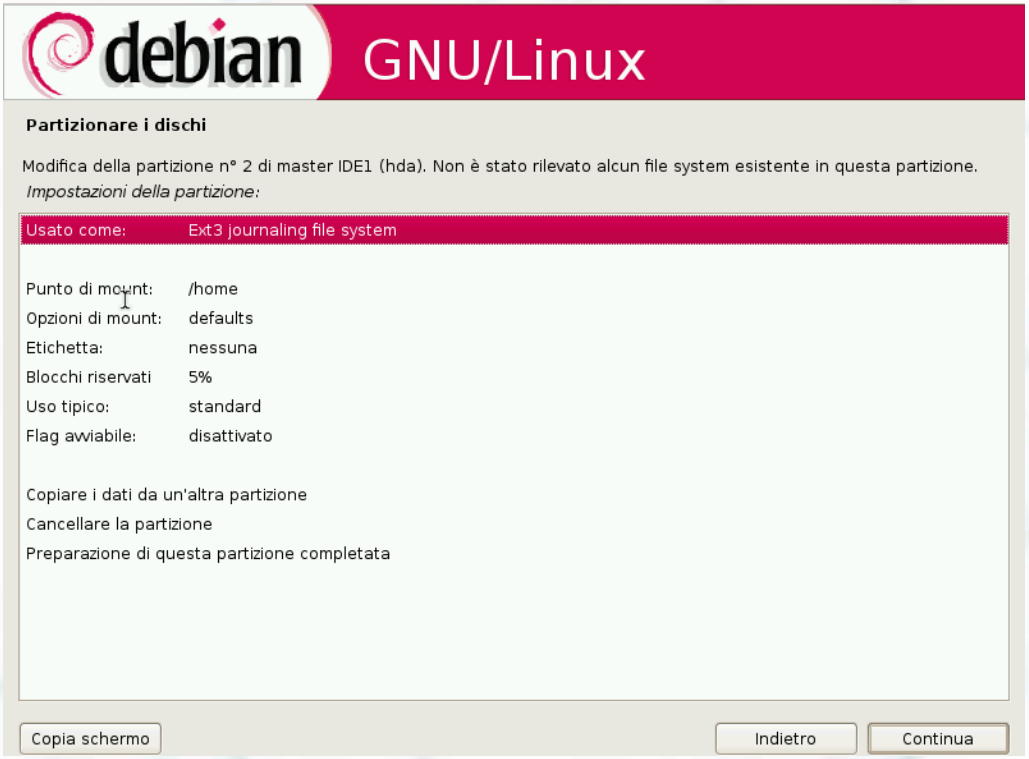


Come vediamo la prima partizione con il file system e il mount point è stato creato, proseguiamo...

Creiamo la partizione di swap

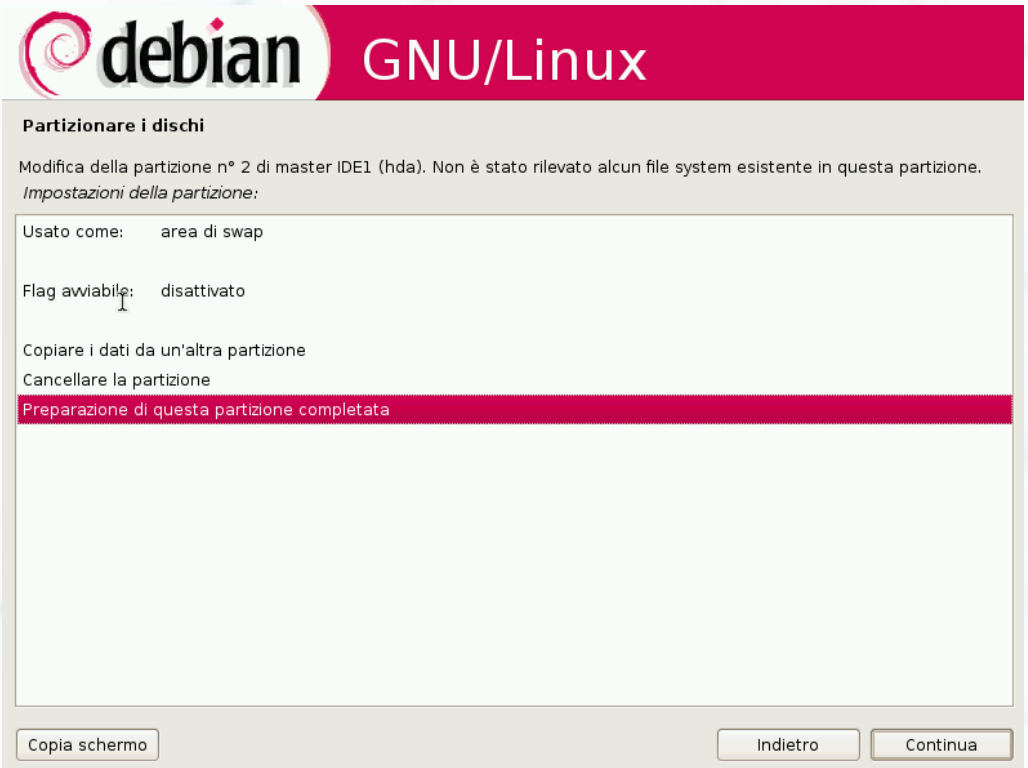


Evidenziamo come vediamo sotto e diamo “invio”

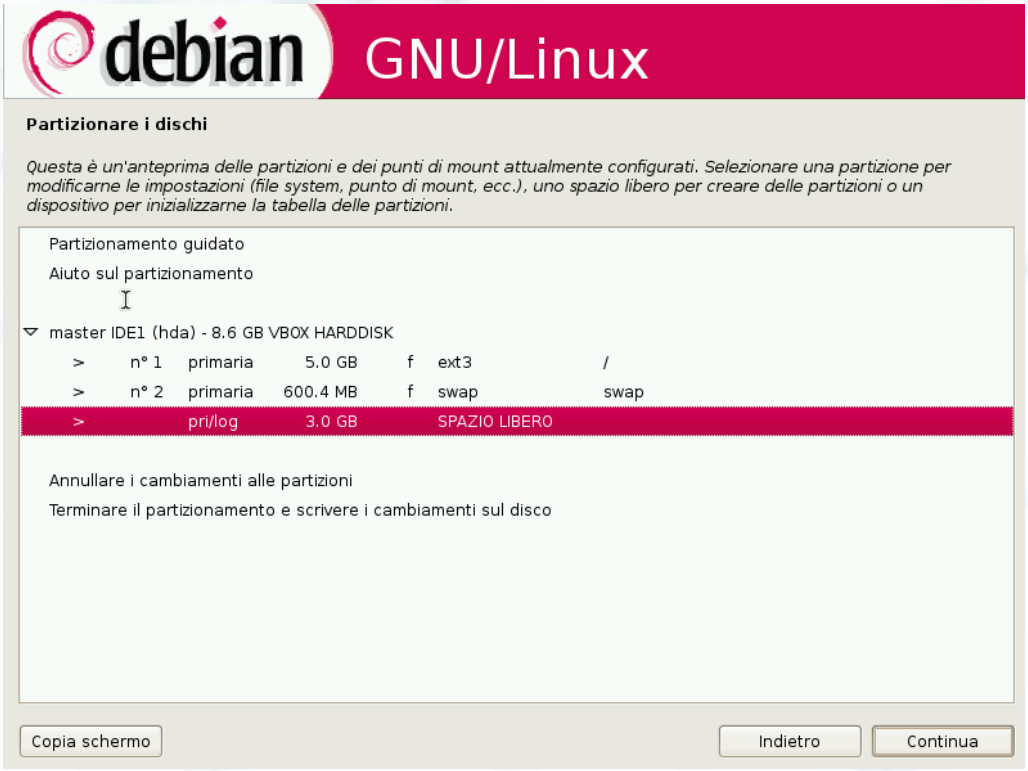


Selezioniamo area di swap

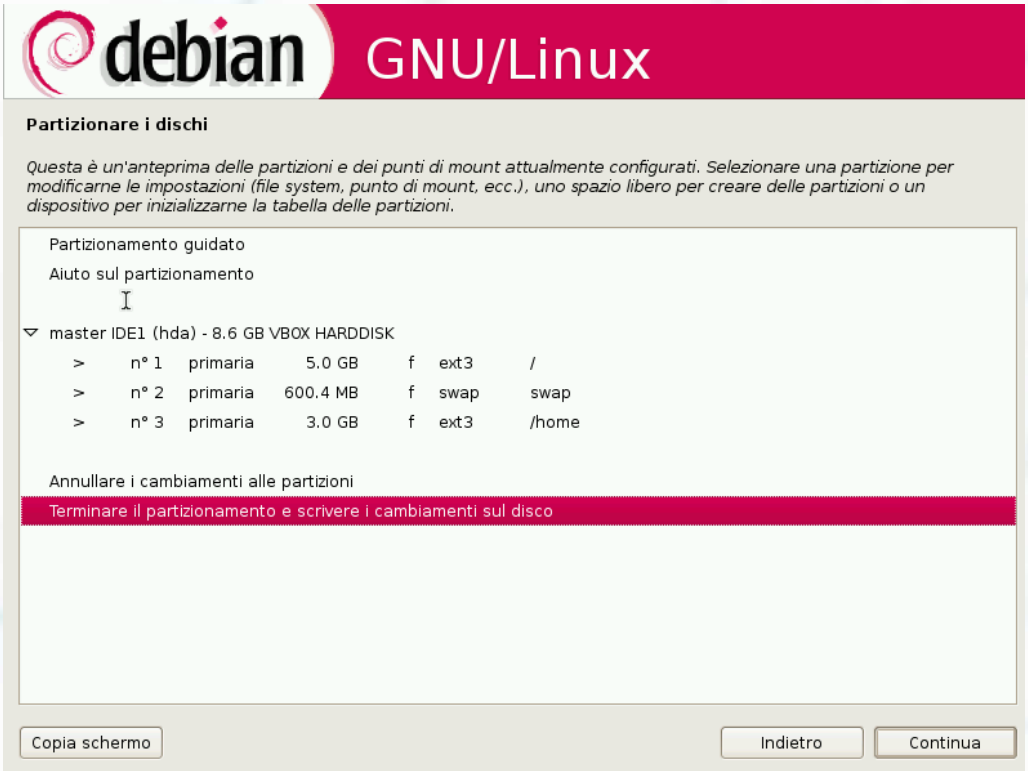
proseguiamo...



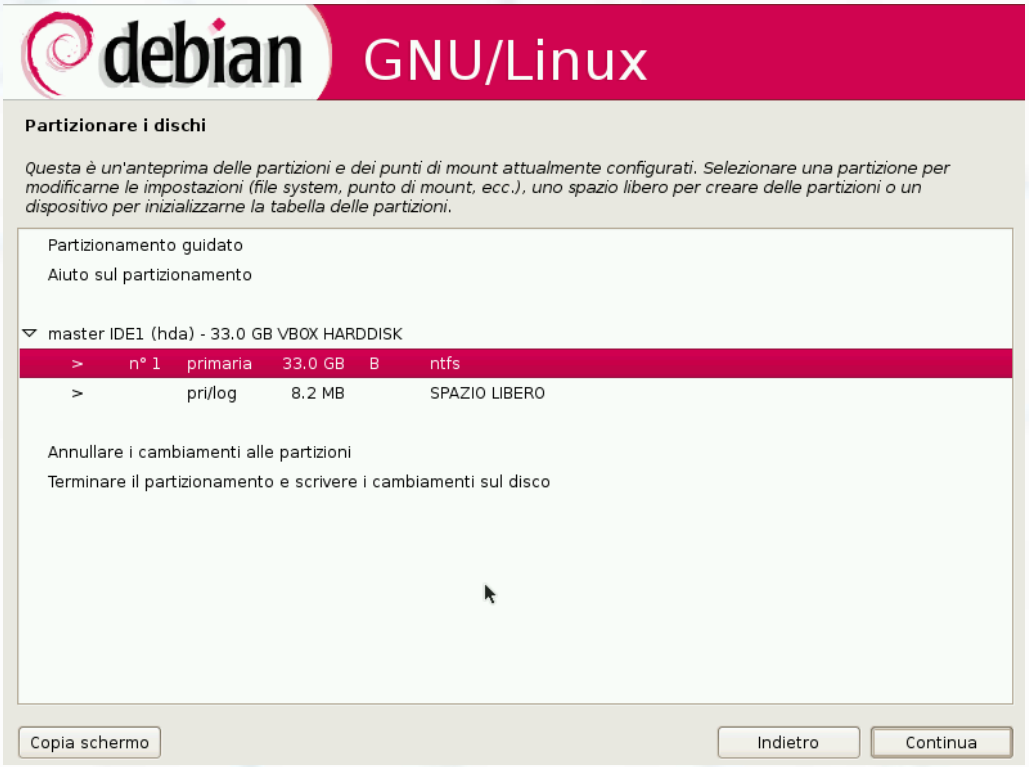
Proseguiamo nella stessa maniera per creare la /home



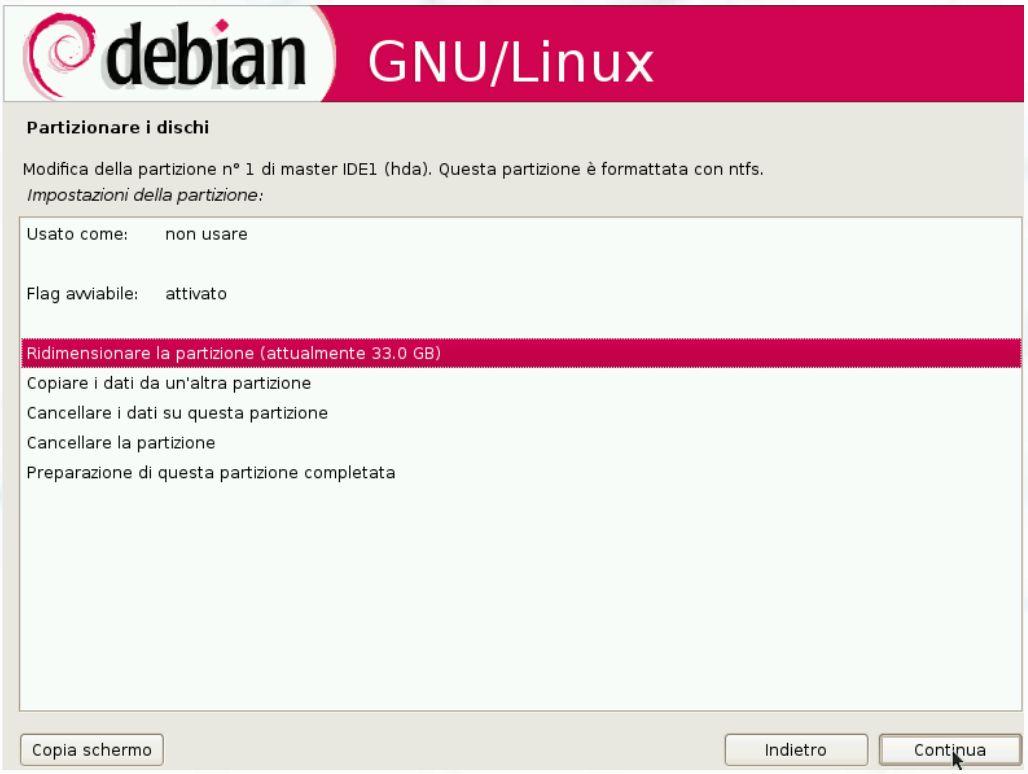
Alla fine avremo qualcosa di simile, se tutto risponde ai nostri desideri proseguiamo



Qua prendiamo in considerazione di avere una partizione con windows e vogliamo restringerla per creare lo spazio per installare debian. Evidenziamo la partizione e diamo "invio"...

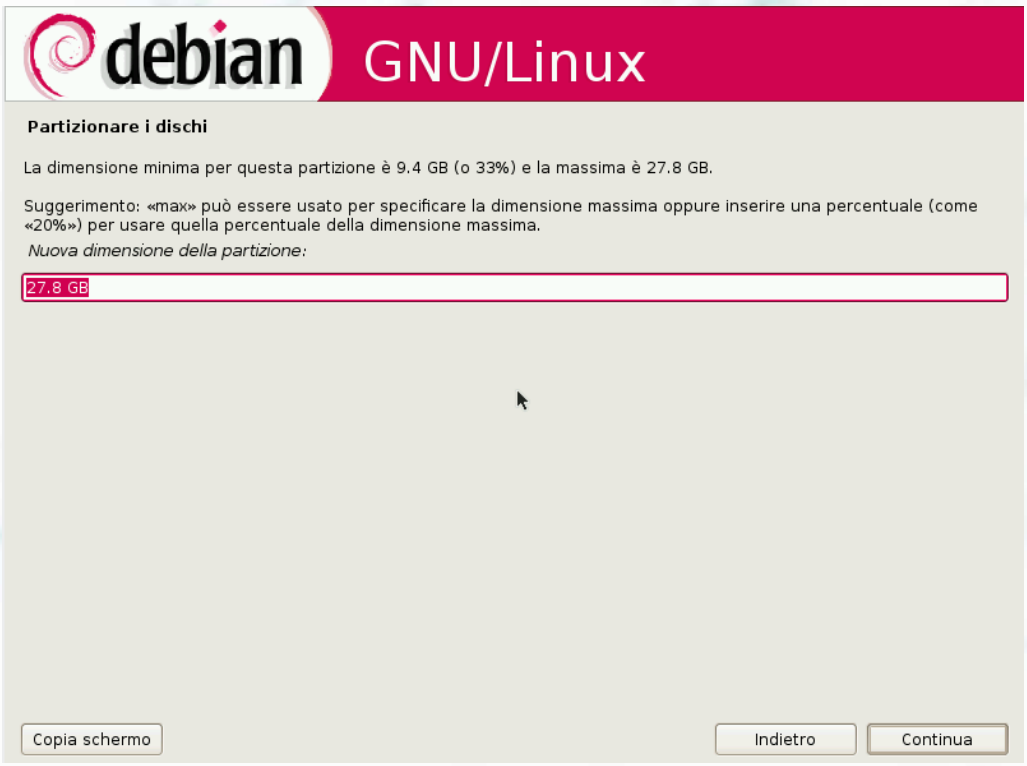


...scegliamo - Ridimensionare la partizione, quindi "invio". Ricordarsi di fare un backup dei dati.



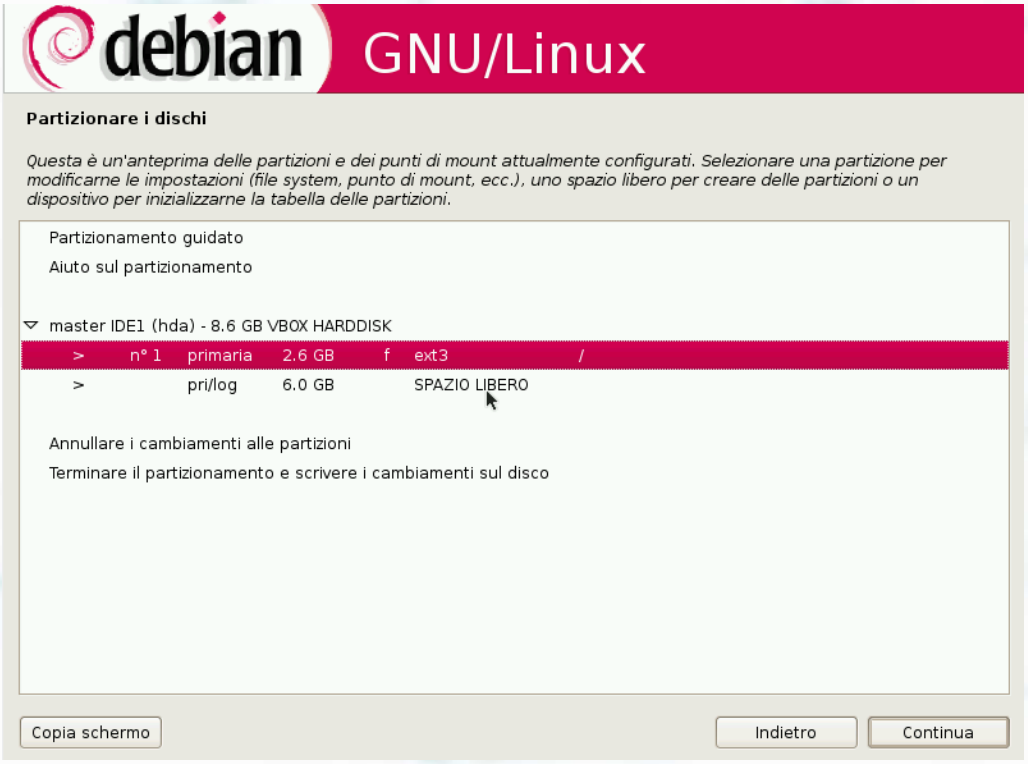
... l'installer ci avvisa del rischio... se siamo sicuri selezioniamo SI...

... l'installer ci propone le possibilità... scegliamo la dimensione che crediamo opportuno ed andiamo avanti... **ATTENZIONE!! fate un backup dei vostri dati, potreste perdere tutto...**



... dopo aver deciso la spazio da liberare, vedremo, come da figura... aspettiamo la fine del processo di ridimensionamento... e alla fine avremo uno spazio libero...

... tipo quello che si vede (ovviamente con dimensioni e file system diversi)... ora selezioniamo lo spazio libero e procediamo alla formattazione come visto precedentemente...



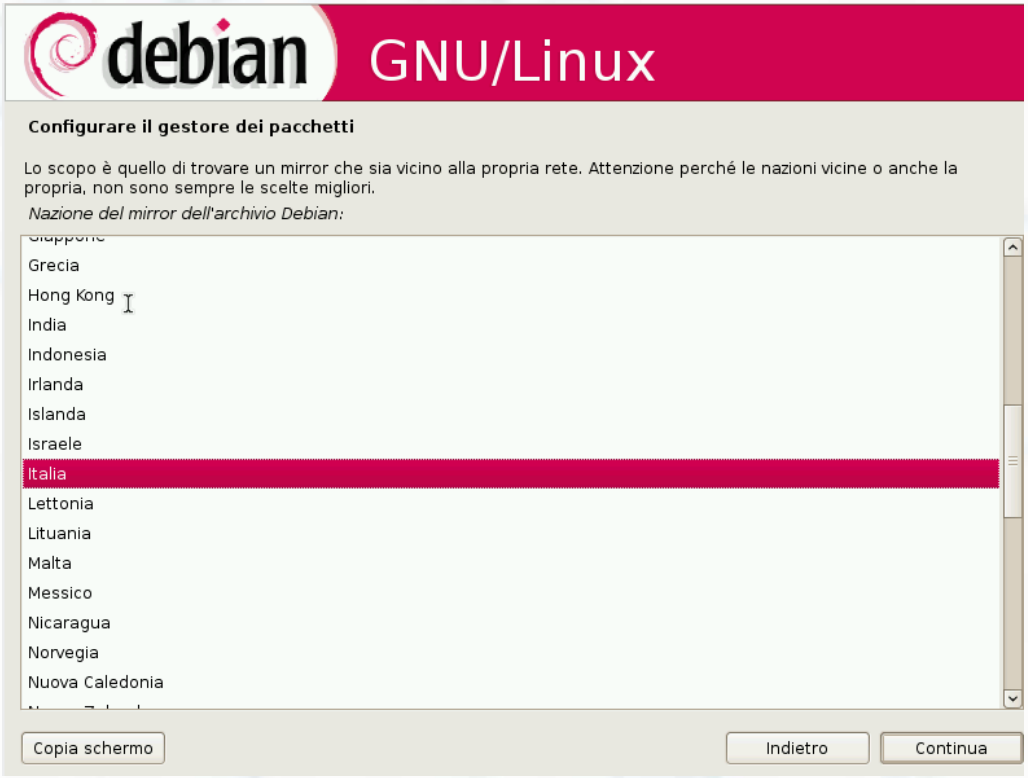
Siamo giunti al punto fatidico, se tutto è come da noi voluto scegliamo si ed andiamo avanti, altrimenti possiamo tornare indietro e fare le dovute modifiche. **Attenzione se si prosegue tutti i dati delle partizioni rimosse varranno distrutti!!**

Si stanno creando le partizioni...

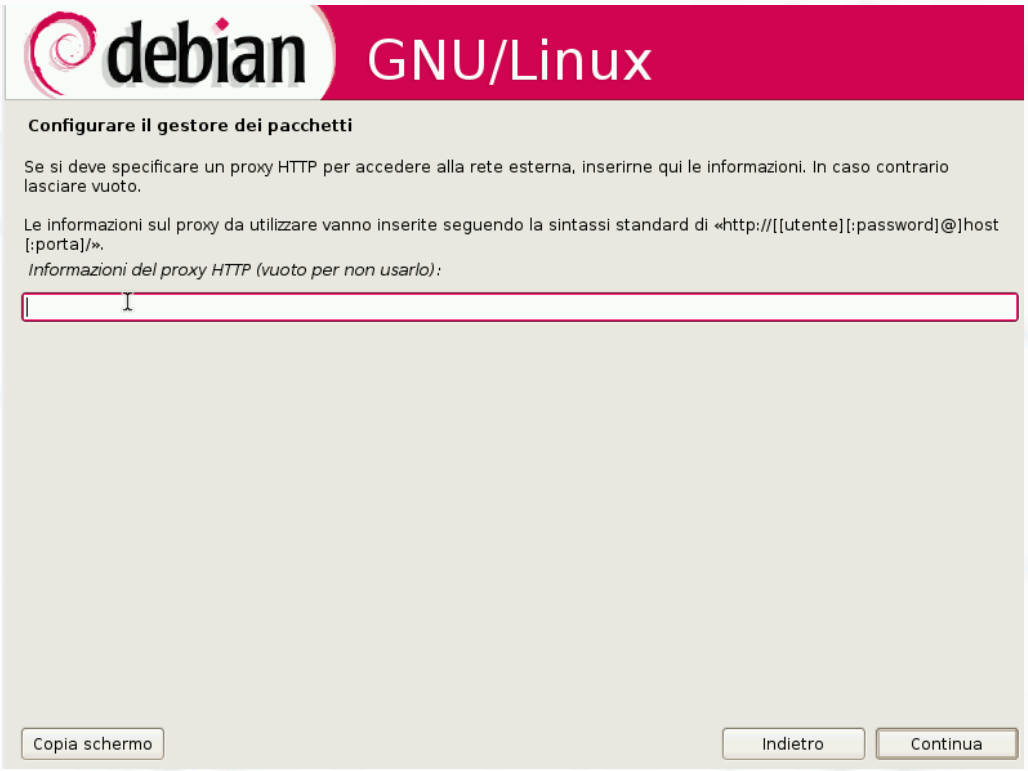
Finito il partizionamento viene installato il sistema base

Installato il sistema base ci viene richiesto l'inserimento della password di /root tasto tab per spostarsi nella seconda riga, dopo ci verrà richiesto il nome e la password dell'user, procediamo allo stesso modo

È giunto il momento della configurazione del mirror da cui scaricare i pacchetti, chi usa il set dei cd o dvd e non ha a disposizione una linea adsl può sorvolare questo passaggio



Scegliamone uno

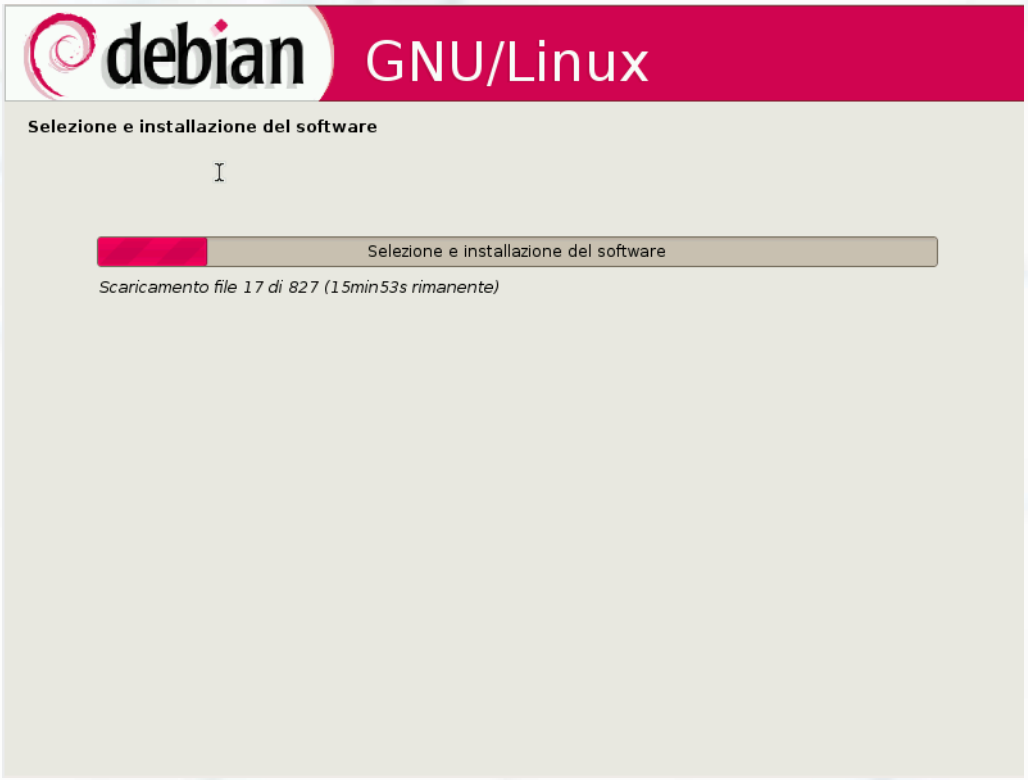


Si sta scansionando il mirror...

Meglio rispondere di sì, alla partecipazione di quali sono i pacchetti più usati...

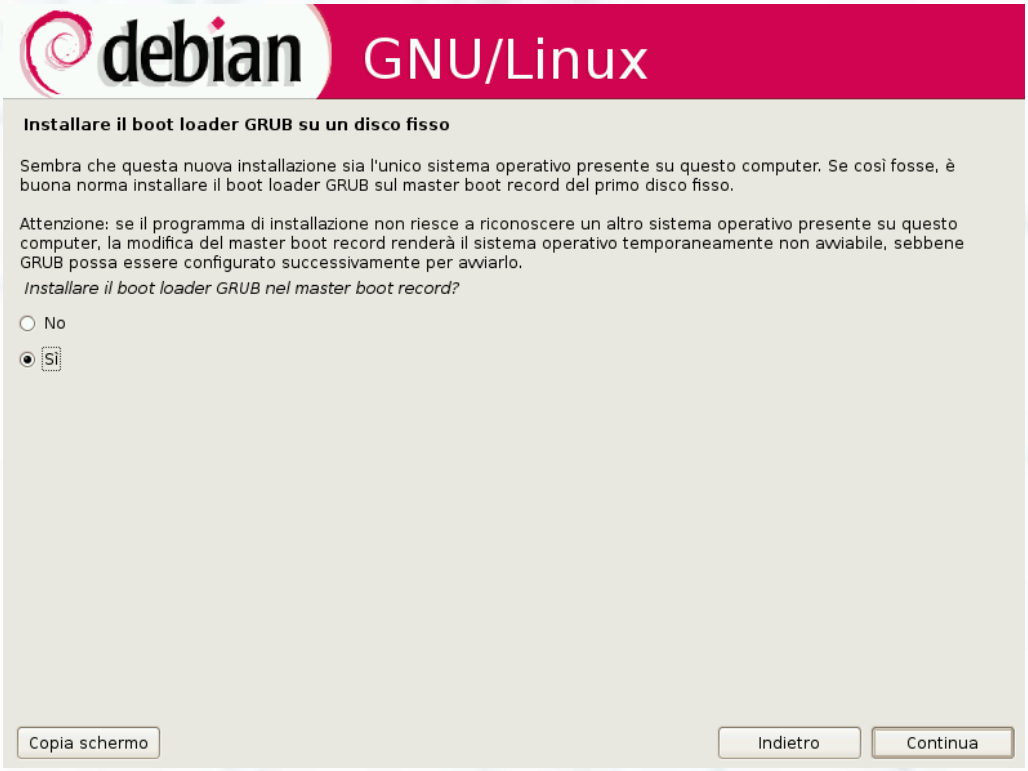
Con la barra spazio mettiamo il segno di spunta per avere il server grafico X, il DE scelto in precedenza e altre opzioni come si vede sotto e procediamo, siamo quasi alla fine...

Inizia lo scaricamento dei pacchetti... rilassiamoci con una bella birra fresca...



Dopo aver scaricato i pacchetti, il tutto viene scompattato...

Siamo arrivati alla conclusione, installare il boot manager GRUB, selezionare SI se lo si vuole nel MBR, NO se lo si vuole nella partizione di /root

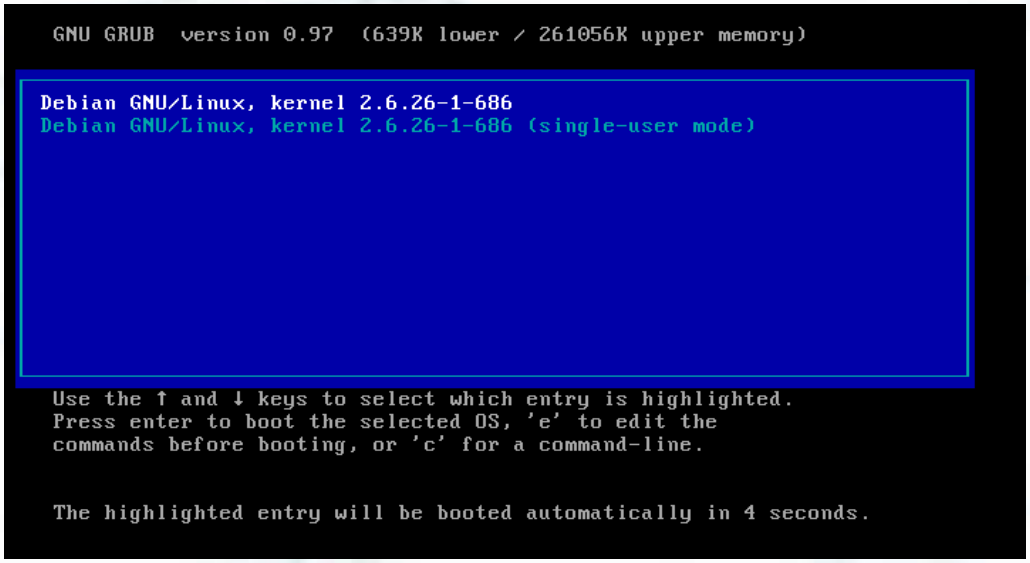


Se non si vuole grub nel MBR scegliamo la partizione dove è /root

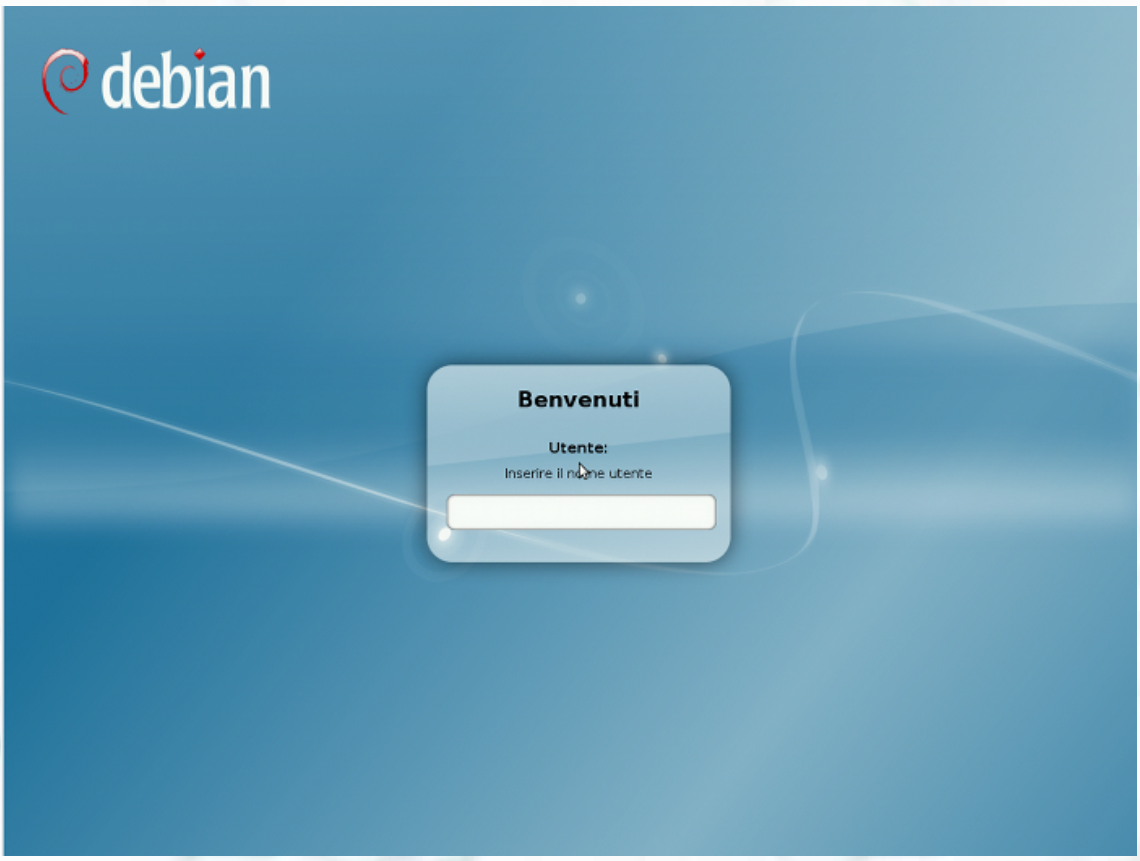
Come da es. /dev/(x)da(x)

Se tutto è andato a buon fine vedremo... togliamo il cd e riavviamo... dopodiché...

... il boot da Hard Disk... e

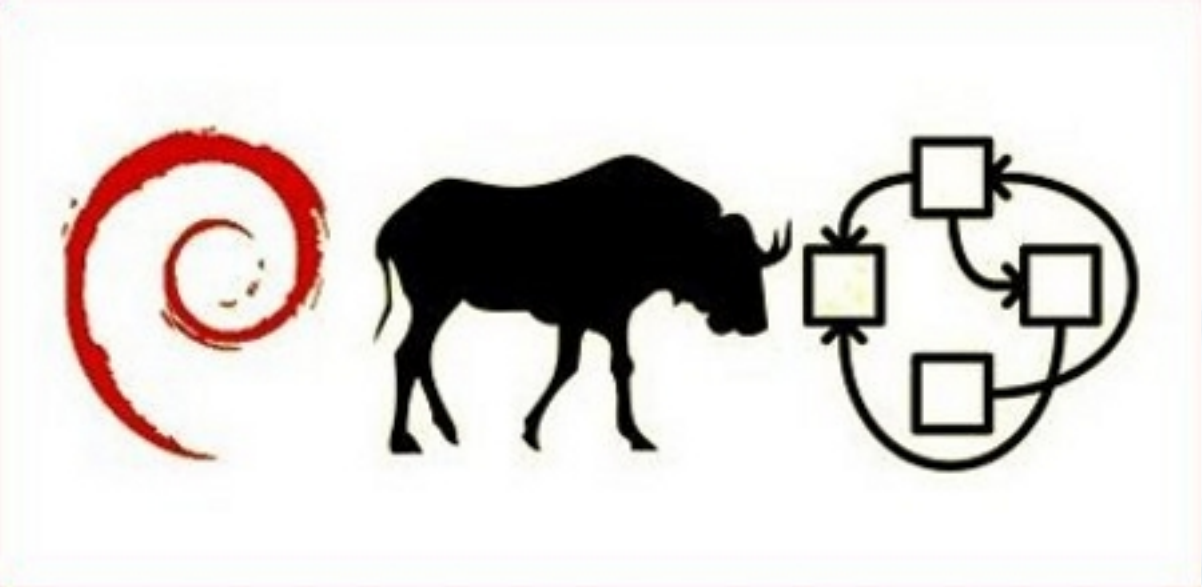


In questo caso GDM ma potrebbe essere KDM o altro... inserire il nome utente e password... ora possiamo usare la nostra debian-lenny... enjoy debian, the most powerful OS on earth!!



Capitolo 4

Debian GNU/Hurd



Conoscevate solo debian GNU/Linux? Vi daremo la possibilità di studiare il vostro sistema operativo preferito con alla base, il sistema sperimentale GNU/Hurd, basato a sua volta sul microkernel Mach.

A seguire una guida sull'installazione di questo sistema operativo, sull'emulatore QEMU

4.1 Installazione di Debian GNU/Hurd su qemu

4.1.1 Installare qemu

Il pacchetto è presente nei repository di tutte le release. Per installarlo usate il vostro gestore di pacchetti preferito. Per l'occasione utilizzeremo aptitude.

```
# aptitude install qemu
```

Installare l'accelerazione kqemu

Per velocizzare l'emulatore è possibile installare un modulo per l'accelerazione. Per compilare il modulo utilizzeremo module-assistant:

```
# m-a a-i kqemu-source
```

Attenzione! Per compilare il modulo è necessario il pacchetto *kqemu-common*; in ogni caso, se non lo abbiamo installato prima, quest'ultimo verrà automaticamente installato durante la compilazione del modulo.
Per caricare il modulo:

```
# modprobe kqemu
```

Se tutto sarà andato a buon fine, al momento di lanciare qemu non otterremo alcun errore; in caso contrario, un messaggio ci avviserà che l'accelerazione non può entrare in funzione.

4.1.2 Installazione partendo da un'immagine predefinita

Grazie a Michael Banck esiste un'immagine per qemu di debian gnu/hurd: si tratta di un file system di 2 Gb, comprendente un'installazione di base con GRUB 0.97, con il network configurato e un utente root senza password. Inizieremo dunque con il download di quest'immagine

```
# wget http://ftp.debian-ports.org/debian-cd/hurd-i386/K16/
                                                debian-hurd-k16-qemu.img.tar.gz
```

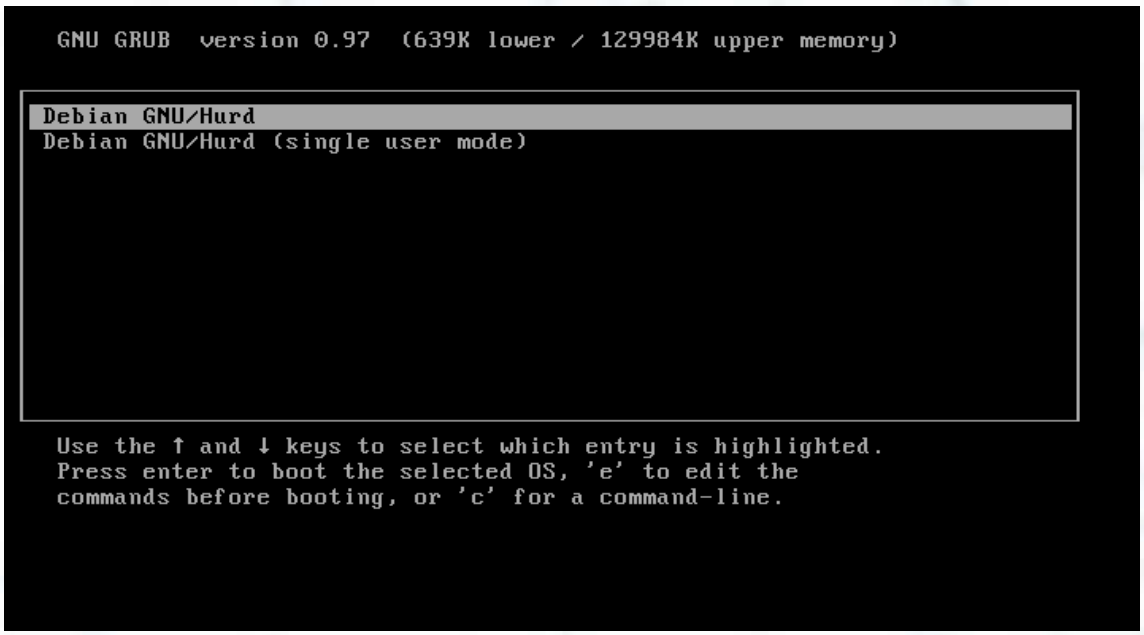
e scompattiamo l'archivio.

```
# tar -xzfv debian-hurd-k16-qemu.img.tar.gz
```

Per far partire la macchina virtuale con la nostra immagine daremo il comando:

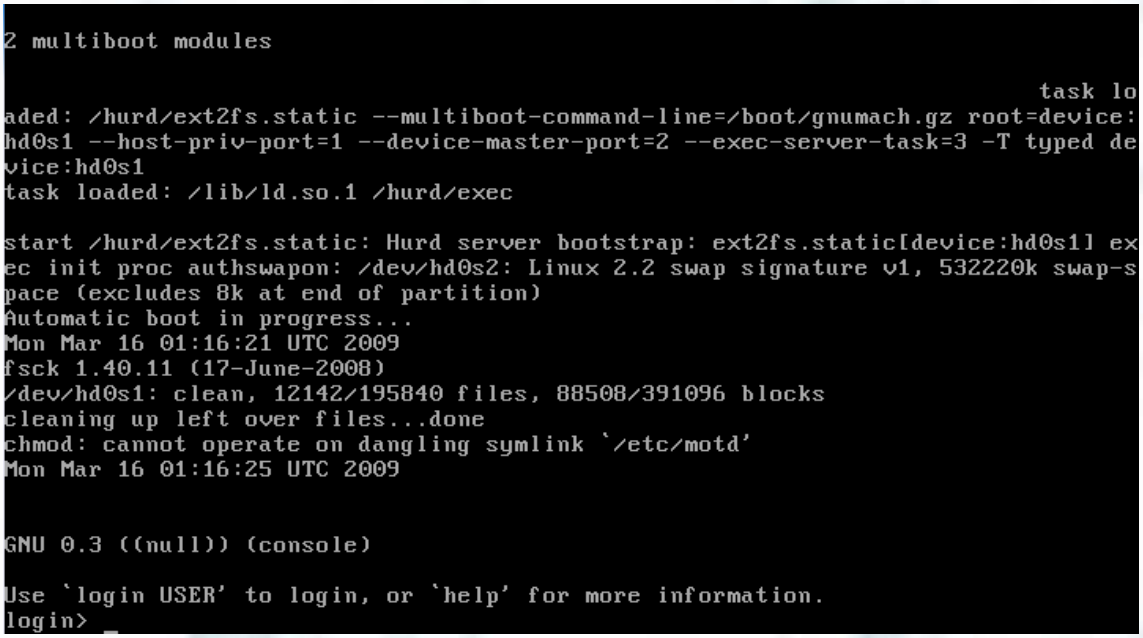
```
$ qemu debian-hurd-k16-qemu.img
```

Apparirà una finestra a video con visualizzato il BIOS della macchina virtuale. A boot avvenuto comparirà la nota schermata di GRUB 0.97, con in selezione la nostra debian GNU/Hurd.



Qualche secondo (o dopo aver premuto il tasto **invio** se la pazienza non è il vostro forte) e potremo assistere al caricamento del sistema.
Il login con l'utente root è leggermente diverso da ciò con cui siamo abituati:

```
login> login root
```



Fatto questo, ci si ritrova in un ambiente più conosciuto: si tratta della console di Hurd. A questo punto, anche se in un sistema un po' primitivo e povero, ci troviamo in debian a tutti gli effetti. Altrimenti detto, utilizzeremo i comandi ai quali siamo abituati da debian GNU/Linux.

I Repository

Andremo ora a sintonizzare il sistema. Come primo punto dovremo "aggiustare" il tiro dei repository.

Battendo i primi comandi nel nuovo sistema noterete un problema con il layout della tastiera: en_US.

Vedremo appena più in basso come ovviare a questo problema, aggiorniamo velocemente il sistema e poi saremo pronti per la configurazione ottimale. Per fare ciò, dovremo solo sapere che la barra " / " si trova al posto del trattino " - " e il trattino si trova al posto dell'apice " ' " (o punto interrogativo " ? "). Essendo debian GNU/Hurd un sistema operativo ancora in fase esperimentale, ci baseremo su sid e su dei ports specifici. Assicuriamoci di avere almeno questi (i repository, come in debian GNU/Linux si trovano in */etc/apt/sources.list*):

```
deb http://ftp.it.debian.org/debian/ sid main
deb http://ftp.debian-ports.org/debian unreleased main
```

Il mirror dei primi repository potrà essere cambiato a seconda della nazione in cui vi trovate. Per modificare il file utilizzeremo un editor di testo qualsiasi. Nel mio caso, ho utilizzato senza problemi "nano".

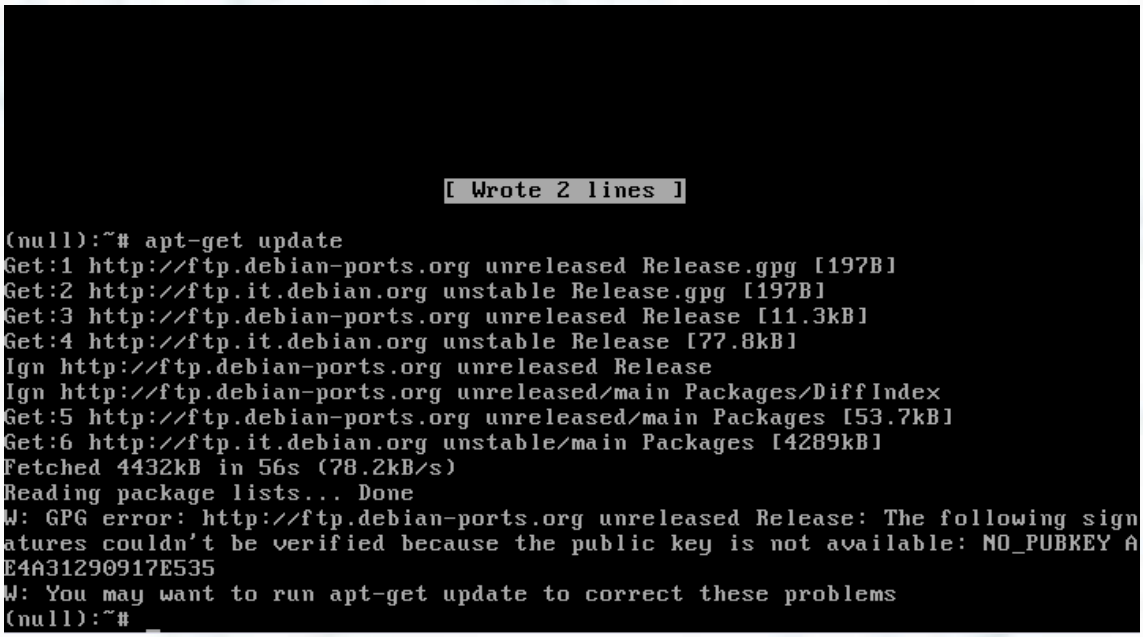
Aggiorniamo il sistema

Utilizzando apt-get. Aptitude sembrerebbe non essere al 100% compatibile con il sistema, a causa di due pacchetti non installabili (*libapt-pkg-libc6.6-6-4.6* e *libcwidget1*).
Per fare ciò, come di consuetudine diamo:

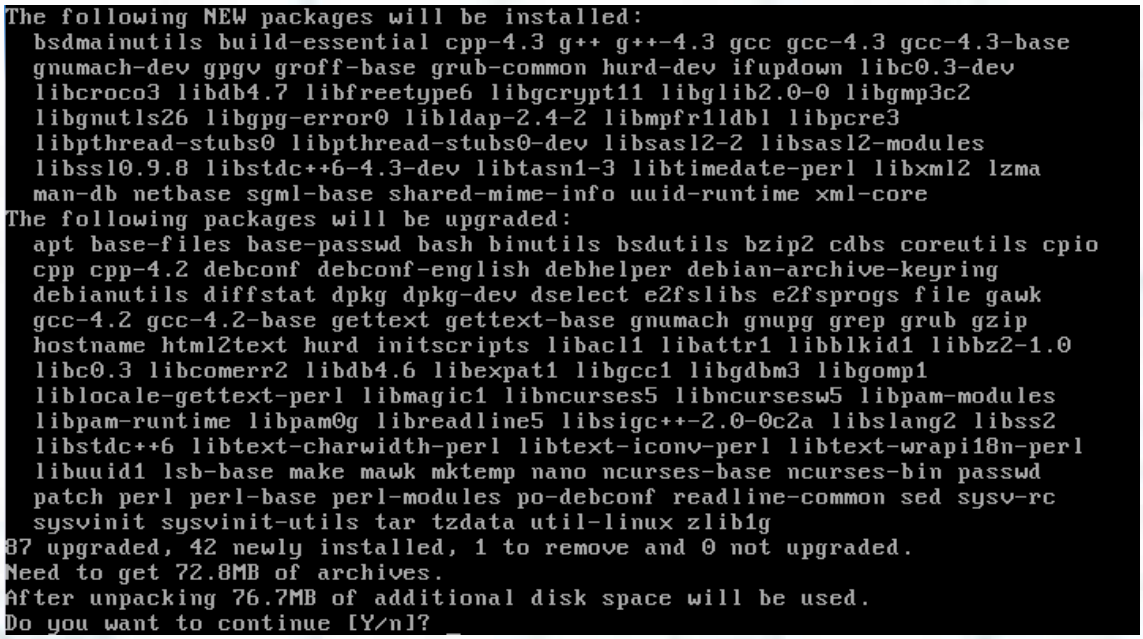
```
# apt-get update

# apt-get dist-upgrade
```

```
# apt-get update
```




```
# apt-get dist-upgrade
```



Aptitude verrà disinstallato a causa dei problemi sopra-citati. Inoltre i *debian-ports* sembrerebbero non avere una chiave pubblica e dovremo confermare l'installazione dei pacchetti che provengono da questo repository. Fatto ciò, abbiamo ora la nostra debian aggiornata.

Il layout della tastiera

Al momento non sembrerebbe esistere un metodo "ufficiale" per ovviare al problema. Esiste però una buona soluzione inventata da Julien Puydt, il quale ha scritto uno script in perl (pacchettizzato per debian) che ci permette di configurare la tastiera. Per scaricare il pacchetto utilizzeremo wget che dovremo prima installare.

```
# apt-get install wget
```

Scaricheremo poi il pacchetto da dei repositories francesi non ufficiali e andremo ad installare il pacchetto con *dpkg*.

```
# wget http://packages.hurdfr.org/experimental/binary-hurd-i386/
                                                    clavier_0.2_hurd-i386.deb
# dpkg -i clavier_0.2_hurd-i386.deb
```

L'installazione porterà i seguenti files nel nostro sistema:

```
/usr/share/clavier/            ; dove troveremo map.fr, map.de, map.en, map.es,
                                 map.dvorak, map.fr-dvorak

/etc/default/keymap            ; dove metteremo la sigla seconda la lingua voluta
                                 (ciò che viene dopo 'map.', di default "fr")

/etc/init.d/clavier            ; ciò che farà partire lo script al boot

/usr/lib/clavier/loadkeys.pl   ; lo script vero e proprio
```

La struttura dei files con la mappatura (*map.**) è molto semplice e possiamo creare dei files noi stessi (o adattare) per tutte le lingue che vogliamo. Si tratta solo di caricare il filesystem sul nostro sistema, aggiungere i caratteri che abbiamo bisogno, cambiare la posizione dei tasti, ed infine semplicemente salvare il file che verrà caricato al boot di qemu. Nel mio esperimento sono riuscito a configurare una tastiera apple praticamente perfettamente, assegnando ad ogni tasto il carattere voluto. Per montare il filesystem sul nostro sistema, si guardi la sezione 4.1.4, "Trasferimento files".
Per far partire lo script all'avvio daremo ancora il comando:

```
# update-rc.d clavier defaults
```

4.1.3 Installazione dal CD originale di debian GNU/Hurd

Se volessimo ora installare debian GNU/Hurd senza un'immagine già pronta procederemo nel modo seguente. Prima di tutto dovremo procurarci i dischi del sistema. Sul server *http://ftp.debian-ports.org/debian-cd/hurd-i386/K16/* fra le varie immagini abbiamo due possibilità per incominciare: sia partendo dal CD1, *debian-K16-hurd-i386-CD1.iso*, sia da una versione ridotta, *debian-K16-hurd-i386-mini.iso*, che rappresenta un po' la classica "netinst" di debian GNU/Linux per un'installazione dalla rete.
Inoltre ci servirà un'immagine floppy di GRUB per avviare il sistema. Per questioni di semplicità utilizzeremo GRUB 0.97, reperibile all'indirizzo *ftp://alpha.gnu.org/gnu/grub/grub-0.97-i386-pc.ext2fs*.
A questo punto abbiamo tutto l'occorrente e potremo iniziare l'installazione.

Creare un'immagine disco per Qemu

Quest'operazione è molto semplice. Diamo il comando:

```
$ qemu-img create nomeimmagine.img XG
```

dove *nomeimmagine* è un nome a caso e X è la grandezza in Gb (G) dell'immagine. Nel mio caso ho creato un'*immagine debian_hurd.img* da 5Gb. Per questioni di spazio utilizzeremo inoltre l'immagine del CD mini; purtroppo non sono però riuscito a trovare un mirror per l'installazione dal network. Negli esempi a seguire ci baseremo su questi nomi.

Avviare la macchina virtuale con l'immagine del CD

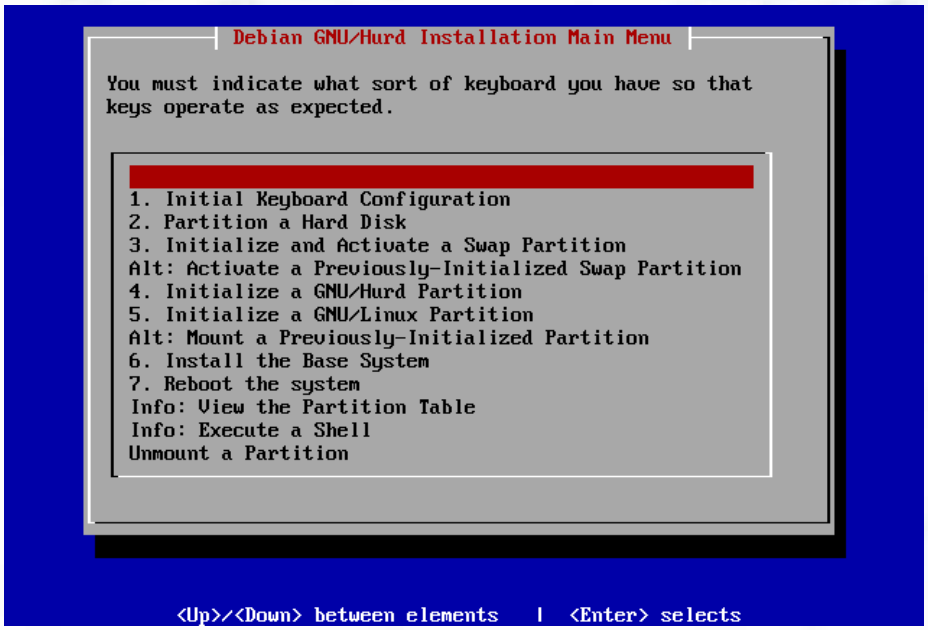
Avvieremo ora la macchina virtuale con il CD. Per fare ciò daremo il comando:

```
qemu debian_hurd.img -cdrom debian-K16-hurd-i386-mini.iso -boot d
```

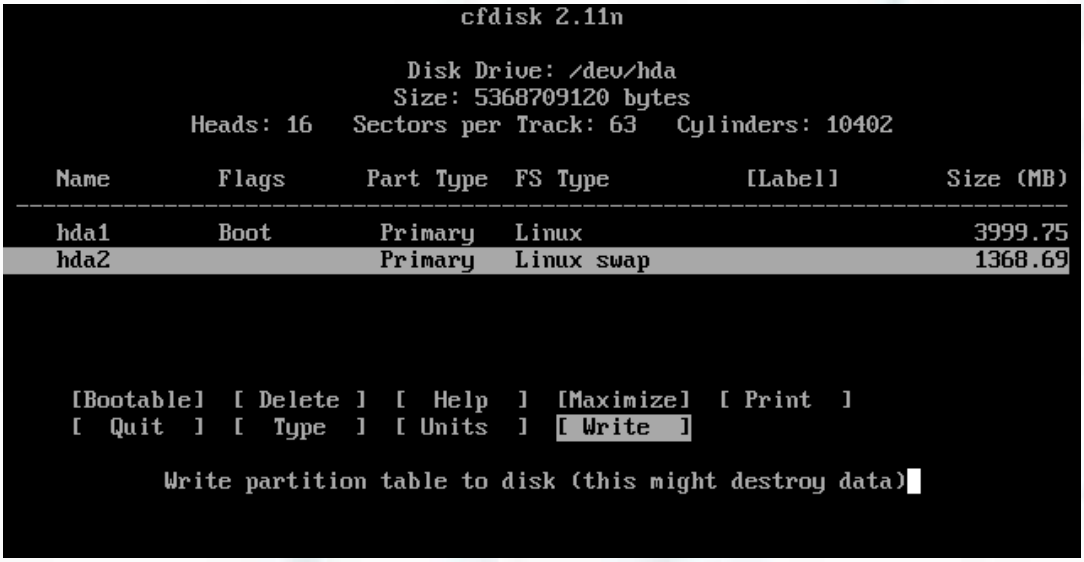
dove l'opzione *-cdrom* ci permette di utilizzare l'immagine *.iso* come un CD vero e priorio e l'opzione *-boot d* ci permette il boot dal CD stesso.

Installazione sistema

L'installer utilizza Linux (`This disk uses the linux kernel 2.4.18-bf2.4 to start installation`) per avviare l'installazione con una comoda interfaccia ncurses. Dopo aver eseguito il boot e visualizzato qualche nota sulla release troveremo la pagina dell'installer vero e proprio.

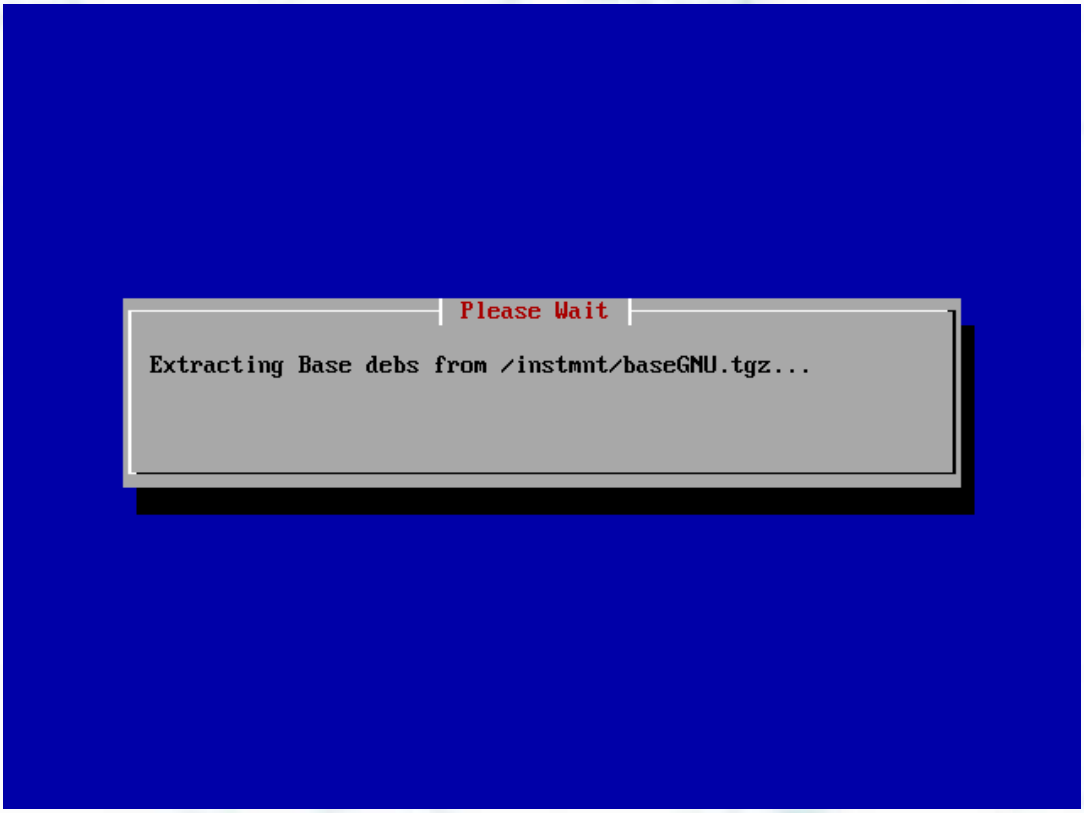


- **Configurazione tastiera**: scegliendo semplicemente la mappatura voluta
- **Partizionamento Hard Disk**: sceglieremo una partizione primaria confermando la riscrittura della tavola delle partizioni, la renderemo "bootabile" e sceglieremo come file system Linux (*codice 83*). Inoltre creiamo una partizione di swap con il resto dello spazio (*codice 82*). Nel mio esperimento ho scelto una partizione primaria di 4Gb e la swap da 1Gb. Probabilmente entrambe esagerate, ma meglio così che il contrario.
- **Inizializzare e attivare una partizione di swap**: confermeremo l'inizializzazione e la partizione verrà montata.



- **Inizializzare una partizione GNU/Hurd**: confermeremo semplicemente il nostro volere (volendo, tralasciando il controllo di *bad-block* sul disco) ed infine monteremo il file system.
- **Inizializzare una partizione GNU/Linux**: per cosa? ;-).
- **Installare il sistema di base**: purtroppo l'installazione dal network sembrerebbe non funzionare. Nonostante la configurazione del network con DHCP funziona perfettamente, non viene trovato un server utilizzabile allo scopo. Sceglieremo dunque l'installazione da cdrom scegliendo come installer la direcotry */insmnt*.

L'installazione del sistema, soprattutto se si è utilizzato il primo CD d'installazione (al posto della versione mini), può durare un momento, anche significativo se si ha un processore di vecchia generazione. Starà a voi scegliere fra una qualche lettura appassionante, o la tradizionale birra ghiacciata ;-).



- **Riavviare il sistema**: infine riavvieremo il sistema da quest'opzione. Al riavvio, quando riapparirà il menu del CD possiamo chiudere qemu

Bisognerà ora configurare GRUB per poter avviare il sistema. Per fare ciò creeremo un file *menu.lst* con la configurazione necessaria:

Creiamo una directory in */mnt* per montare GRUB:

```
# mkdir /mnt/grub
```

Montiamo l'immagine floppy di GRUB in questa directory (proprio perchè di un'immagine si tratta con l'opzione *-o loop*):

```
# mount -o loop grub-0.97-i386-pc.ext2fs /mnt/grub
```

Creiamo il file menu.lst:

```
# nano /mnt/grub/boot/grub/menu.lst
```

Inseriamo la seguente configurazione:

```
title debian GNU/Hurd
kernel (hd0,0)/boot/gnumach.gz root=device:hd0s1
module (hd0,0)/hurd/ext2fs.static --multiboot-command-line=${kernel-command-line}
                --host-priv-port=${host-port} --device-master-port=${device-port}
                --exec-server-task=${exec-task}  -T typed  ${root} $(task-create)
                  $(task-resume)
module (hd0,0)/lib/ld.so.1 /hurd/exec $(exec-task=task-create)


title debian GNU/Hurd (single user mode)
kernel (hd0,0)/boot/gnumach.gz root=device:hd0s1 -s
module (hd0,0)/hurd/ext2fs.static --multiboot-command-line=${kernel-command-line}
                --host-priv-port=${host-port} --device-master-port=${device-port}
                --exec-server-task=${exec-task} -T typed  ${root}  $(task-create)
                  $(task-resume)
module (hd0,0)/lib/ld.so.1 /hurd/exec $(exec-task=task-create)
```

dove con l'opzione *-s* andremo ad avviare il kernel in modalità *single user mode* per configurare hurd e con le altre due opzioni *module* andremo ad avviare i server *exec* e del filesystem di root.

ATTENZIONE:
i comandi relativi ai moduli devono essere scritti su una riga sola, senza andare a capo.

Dopo aver salvato, smontiamo l'immagine di GRUB:

```
# umount /mnt/grub
```

Riavvieremo ora l'immagine disco con il sistema operativo dal floppy (virtuale) con l'immagine di GRUB:

```
qemu debian_hurd.img -fda grub-0.97-i386-pc.ext2fs -boot a
```

dove *-fda* ci permette di utilizzare il file di GRUB come un floppy e *-boot a* darà l'avvio da quest'ultimo.
Una volta ripartito il sistema avremo l'immagine di GRUB con il nostro GNU/Hurd disponibile nel menu. Lanceremo il boot nella modalità *single user mode*. Una volta finito ciò digiteremo il comando per definire il terminale:

```
# export TERM=mach
```

E andremo poi ad abilitare di default la console di Hurd in */etc/default/hurd-console.dpkg-new*, modificando la linea:

```
# set this to 'true' to run the hurd console on bootup
ENABLE='false'
```

da *false* a *true*.

Infine facciamo partire l'installazione:

```
./native-install
```

N.B.: Nella selezione del tipo di terminale potreste incontrare un fastidioso problema: con il layout americano non si arriva a digitare il simbolo“ = ”. Io ho risolto montando la immagine di qemu nel sistema (vedi *Trasferimento files*, sezione 4.1.4) e creando una directory in /home chiamata proprio “ = ”. Così dalla shell di Hurd sarà possibile richiamare questa directory grazie al tabulatore, cancellare ciò che ci sta attorno e completare il comando.
A questo punto, seguendo le istruzioni daremo il comando

```
# reboot
```

riavviando il sistema (questa volta in modalità normale) ed infine ancora

```
./native-install
```

e saremo finalmente a cavallo!
Dal prossimo riavvio accederemo finalmente alla console di Hurd ed avremo praticamente terminato l'installazione.

Configurare la rete

Per configurare ora la rete, prendendo i parametri dell'emulazione di QEMU, utilizzeremo *pfinet*:

```
settrans -fgap /servers/socket/2 /hurd/pfinet -i eth0 -a 10.0.2.15 -g 10.0.2.2    \
         -m 255.255.0.0
```

Bisognerà poi scrivere il nameserver 10.0.2.3 nel file */etc/resolv.conf*:

```
echo "nameserver 10.0.2.3" > /etc/resolv.conf
```

A questo punto non ci resta che collaudare la rete con un bel *apt-get update && apt-get dist-upgrade* (dopo aver però dato un'aggiustata ai repository *[v. sopra]*) e la debian gnu/hurd è installata!

Attenzione! Dopo il primo aggiornamento il file */etc/default/hurd-console* verrà modificato. Dovremo ricordarci d'inserire l'opzione `ENABLE true` per avviare la console al prossimo boot.

Attivare la swap e configurare /etc/fstab

Dopo il primo avvio noteremo che la swap non viene avviata. Per fare ciò dovremo dapprima creare un nuovo *device* (dopo esserci loggati come *root*):

```
# cd /dev ; MAKEDEV hd0s2 hd2
```

Fatto ciò andremo a modificare il nostro */etc/fstab* inserendo la partizione di swap e una riga per montare il cdrom. Una volta aperto il file con qualsiasi editor aggiugeremo le seguenti linee:

```
/dev/hd0s2 none    swap sw 0 0
/dev/hd2  /cdrom    iso9660fs  ro,noauto  1 1
```

Al prossimo reboot verrà così caricata la swap (oppure da subito dando da root il comando *swapon -a.*

Aggiornare il sistema ed installare X

Grazie ad uno script contenuto nel primo CD d'installazione possiamo aggiornare il sistema con dei pacchetti importanti per il sistema e installare un ambiente grafico. Visto che abbiamo scaricato l'immagine disco da 50Mb (mini), onde evitare di dover scaricare il CD1 solo per questi due script ve li riporto in questa guida.

Aggiornare il sistema (script *install.sh*):

```
#!/bin/bash
apt-get update
apt-get dist-upgrade
sync
echo ""
echo This script installs the required, important and standard packages
echo ""
apt-get install --fix-missing coreutils ncurses-bin libncurses5 tar               \
                libtext-iconv-perl libtext-charwidth-perl libtext-wrapi18n-perl \
                mawk libattr1 libacl1 libss2 e2fslibs e2fsprogs libuuid1        \
                libcomerr2 libblkid1 base-files bash base-passwd less apt-utils \
                dialog
sync

apt-get install --fix-missing hostname hurd grep libgcc1 libstdc++6 gcc-4.2-base  \
                libc0.3 gzip libdb4.3 debianutils dpkg dselect diff debconf     \
                debconf-i18n zlib1g lsb-base passwd libslang2 sed               \
                sysvinit-utils initscripts sysv-rc sysvinit

sync

apt-get install --fix-missing  liblocale-gettext-perl findutils perl-base         \
                libpam0g libpam-runtime libpam-modules bsdutils ncurses-base    \
                tzdata libsepol1 mktemp util-linux bsdutils util-linux          \
                libsasl2-2 cron cpio netbase
sync

apt-get install --fix-missing libnewt0.52 whiptail libncursesw5 tcpd libwrap0     \
                tasksel-data tasksel info libtasn1-3 libsigc++-2.0-0c2a man-db  \
                update-inetd vim-common wget adduser aptitude apt ed libbz2-1.0 \
                bsdmainutils libgnutls13
sync

apt-get install --fix-missing libgdbm3 groff-base debian-archive-keyring libdb4.4 \
                libgcrypt11 libgpg-error0 logrotate laptop-detect liblzo2-2     \
                readline-common libreadline5 libopencdk10 libssl0.9.8 libldap2  \
                libpopt0 nano netcat manpages traceroute vim-tiny anacron
```

Installare ambiente grafico (script *gui.sh*):

```
#!/bin/bash


echo "    Installing x-window-system"

apt-get --fix-missing install xorg xfs xserver-xorg-core twm xfonts-base xnest    \
          xspecs xterm xvfb xfonts-100dpi xfonts-75dpi xfonts-scalable xutils     \
          aterm groff xserver-xorg-video-vesa
```

Per installare gli script possiamo ad esempio creare due files con questo contenuto e metterli nella */home* di debian gnu/hurd, caricando l'immagine di qemu sul nostro sistema (vedi *Trasferimento files*, sezione 4.1.4). Per lanciare gli script daremo semplicemente il comando:

```
# sh install.sh
# sh gui.sh
```

Non ci resta che installare ora un desktop o un window-manager, così come magari un display manager. Al momento ho personalmente qualche problema con *xinit* e *xmodap* da risolvere. Teoricamente sembrerebbe però possibile installare un sistema grafico.
A questo punto dovremo solo installare grub nel **MBR** (Master Boot Record) del nostro disco virtuale per poter avviare il sistema senza aiuto del floppy (virtuale).
Per fare ciò dovremo dapprima copiare il contenuto di grub del floppy nella partizione di boot dell'immagine disco virtuale (montando le rispettive immagini nel sistema gnu/linux, vedi *Trasferimento files*, sezione 4.1.4). Fatto ciò avvieremo il sistema come di consueto (dunque con il boot dal floppy con grub). Arrivati all'interfaccia grafica di grub, anzichè avviare il sistema premiamo " **c** " per avviare la shell di grub. Per installarlo nel MBR digiteremo ora:

```
grub> root (hd0,0)
grub> setup (hd0)
```

per selezionare il *root device* di grub nel MBR e con il secondo comando scrivere il programma in quella parte della partizione primaria. Potremo ora chiudere qemu e al prossimo riavvio, digitando semplicemente:

```
$ qemu debian_hurd.img
```

Non ci resta ora che sperimentare la nostra debian con il nuovo kernel!

4.1.4 Trasferimento files

Per passare files e directories dal nostro sistema di base (debian GNU/Linux) all'immagine disco di qemu, è possibile montare quest'ultima immagine come un disco vero e proprio e scambiare i dati come si farebbe con uno Stick USB. L'immagine disco è suddivisa nel seguente modo:

```
# fdisk -ul debian-hurd-k16-qemu.img
...

Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000000

                  Device Boot   Start       End    Blocks   Id  System
debian-hurd-k16-qemu.img1   *       63   3128831   1564384+  83  Linux
debian-hurd-k16-qemu.img2      3128832   4193279    532224   82  Linux swap/Solaris
```

Come possiamo vedere, la partizione di root non incomincia da 0, ma da 63. Per montare la parte della partizione interessata bisogna calcolare dunque l'offset della stessa. Sapendo che ogni settore è formato da 512 bytes e che abbiamo 63 settori prima dell'inizio della partizione, l'offset sarà calcolato a:

```
63 * 512 bytes = 32256 bytes
```

Per montare ora la partizione, previa la creazione di una directory in /media, daremo il comando:

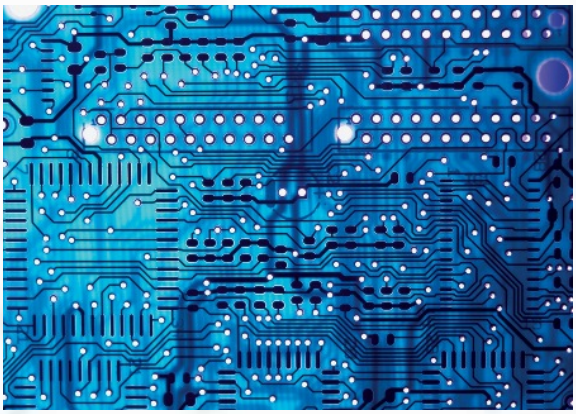
```
# mount -o loop,offset=32256 debian-hurd-k16-qemu.img /media/qemu_hurd
```

Link utili

http://www.gnu.org/software/hurd/hurd/running/qemu.html
http://www.gnu.org/software/hurd/users-guide/using_gnuhurd.html
http://www.gnu.org/software/grub/manual/html_node/Configuration.html
http://www.gnu.org/software/grub/manual/html_node/Installing-GRUB-natively.html
http://www.debian.org/ports/hurd/hurd-install
http://www.debian.org/ports/hurd/hurd-cd
http://www.hurdfr.org/pages/doc/GuideInstall.pdf
http://wiki.hurdfr.org/index.php/GNU/Hurd_Internals_Guide
http://uwhug.org.uk/index.pl?Hurd_Installation_Guide
http://openendeavors.blogspot.com/2008/05/playing-with-debian-gnuhurd.html
http://magnux.free.fr/hurd/cuisine_hurd.html
http://archives.devshed.com/forums/unix-106/en-keyboard-layout-2421684.html
http://salahuddin66.blogspot.com/2007/04/install-debian-gnuhurd.html
http://freebox.blogdns.com/index.php/2007/09/04/proviamo-gnuhurd-su-qemu/

Capitolo 5

Hardware & Debian



Tutte le informazioni, esperienze, tuning sull'hardware e debian.
Debian, così come Linux, è una della distribuzioni più versatili adattandosi a quasi ogni tipo di interfaccia possibile.
Ogni tanto è però necessario penare un momento prima di riuscire a configurare il nostro sistema operativo per un certo tipo di hardware.
In questa sezione troverete i nostri esperimenti, così come consigli e hack.

A seguire una guida sull'installazione della nostra debian sull'ASUS eeePc.

5.1 Debian Lenny su ASUS eeePC 900A

Questa guida mostra come è possibile installare e configurare la nostra distribuzione GNU/Linux preferita sul *netbook* per eccellenza: l'**ASUS eeePc**.

Cos'è un netbook? *Wikipedia* recita:

"Il Netbook è un computer mobile minimale ed essenziale, destinato soprattutto alla navigazione in Internet e videoscrittura e pensato soprattutto per un pubblico non professionale."
Potendo contare sulla potenza e sulla versatilità della nostra Debian, trasformeremo questo netbook in un notebook in miniatura, hardware permettendo!

5.1.1 Installazione

Il progetto **debianeeePC** [1] mette a disposizione una piccola immagine avviabile (solo 16 MB) che contiene il debian-installer di Lenny, personalizzato per l'eeePC. [2]
Questo installer permette di scaricare tutto via rete, pertanto dovremo avere a disposizione un access point oppure un router ethernet. E' necessario poter disporre di una memoria USB vuota, che dovremo formattare per fare posto all'intaller e un computer (con GNU/Linux ovviamente!) con cui fare le operazioni preliminari. Per questo possiamo utilizzare l'eeePC stesso che viene fornito con una versione di Xandros GNU/Linux.
Prepariamo la chiavetta USB. La inseriamo e verifichiamo a quale device corrisponde con:

```
$ dmesg|tail
```

Identificato il dispositivo (quello corrispondente all'intero volume e non a una sua partizione), trasferiamo l'installer sulla chiavetta:

```
# dd if=debian-eeepc.img of=/dev/sdX ;sync
```

[1] http://eeepc.debian.net/
[2] http://eeepc.debian.net/debian/images/debian-eeepc.img

Adesso non resta che riavviare l'eeePC con inserita la chiavetta USB, salutando per l'ultima volta Xandros. All'avvio premiamo il tasto F2 per entrare nella configurazione del BIOS dove attiviamo come primo dispositivo di boot la chiavetta USB ed attiviamo pure la scheda wireless, se non lo abbiamo già fatto in precedenza. Il sistema riavviato fa boot direttamente sulla chiavetta USB che abbiamo preparato, senza bisogno di ulteriori azioni. A questo punto ci si presenta agli occhi il classico installer di Debian e procediamo come da nostra abitudine. Consigliamo a questo proposito l'articolo *Installazione grafica di Debian Lenny 5.0* presente in questa rivista (sezione 3.1).

Alcuni consigli:

- sia la scheda wireless che quella ethernet vengono rilevate correttamente e sono entrambe utilizzabili, grazie ai moduli del kernel fornito con l'installer. La connessione wireless può essere instabile, quindi se possibile è preferibile utilizzare l'ethernet.
- in fase di partizionamento si può non riservare nessuna area di swap considerando che 1GB di RAM sia più che sufficiente per un normale utilizzo dell'eeePC.
- Utilizzare un file system ext3 è la scelta migliore: i vantaggi del journaling valgono molto di più della piccola percentuale in più di scritture su disco che questo comporta: il disco a stato solido durerà comunque degli anni.
- Il kernel più adatto per il nostro processore è il *-686*

ATTENZIONE : è possibile che GRUB venga installato sul device sbagliato! Una verifica può evitare spiacevoli inconvenienti. Prima di confermare l'installazione di GRUB, si apre una shell con alt+F2 e si controlla l'output di

```
df -a
```

Dobbiamo verificare che il device montato in */target* sia effettivamente il disco interno, piuttosto che la chiavetta USB. Se non è così prendiamo nota del device corrispondente al disco e installiamo grub su questo. Al primo riavvio modificheremo grub opportunamente per poter avviare correttamente il sistema.
Arrivati alla scelta dei pacchetti da installare è conveniente selezionare solo *Sistema standard* e *computer portatile* per installare tutto ciò di cui abbiamo bisogno (e solo quello) dopo il primo riavvio.

5.1.2 Configurazione

Se tutto è andato a buon fine, al primo avvio ci ritroveremo al login testuale. Entriamo come root e ci accingiamo a completare l'installazione.
L'errore che abbiamo notato in fase di boot:

```
Error: Driver 'pcspkr' is already registered, aborting...}
```

lo risolviamo semplicemente blacklistando il modulo pcspkr, assolutamente superfluo, dato che l'eeepc non è dotato di speaker interno. Al file:

```
/etc/modprobe.d/blacklist
```

aggiungiamo

```
blacklist pcspkr
```

Rete

```
01:00.0 Ethernet controller: Attansic Technology Corp. L1 Gigabit Ethernet Adapter
                                                                          (rev b0)
02:00.0 Ethernet controller: Atheros Communications Inc. AR242x 802.11abg Wireless
                                                        PCI Express Adapter (rev 01)
```

I moduli **Madwifi** sono stati installati automaticamente e la scheda wireless è gia utilizzabile, il modulo *atl1e* , necessario alla scheda ethernet è presente nel kernel 2.6.26, quindi possiamo scegliere di collegarci in rete sia ethernet (*eth0*) che wireless (*ath0*) senza bisogno di intervento alcuno. Il sistema cercherà automaticamente di utilizzare lo stesso metodo utilizzato in fase di installazione.

I repository

Ci troviamo già impostati quelli ufficiali di Lenny e quelli del progetto DebianEeePC per Lenny che contengono molti pacchetti utili, personalizzati per il nostro eeePc. Possiamo aggiungerne eventualmente di personali (es. debian-multimedia).

Server X

Questo modello è equipaggiato con un **display** da 8.9 WSVGA ad una risoluzione di 1024x600 pixel e la **scheda grafica** è la seguente:

```
00:02.0 VGA compatible controller: Intel Corporation Mobile 945GME Express
                                   Integrated Graphics Controller (rev 03)
```

Per una installazione minimale di xorg (80 MB circa) è sufficiente:

```
# aptitude install -R xserver-xorg-video-intel libgl1-mesa-dri xinit
```

Tutto il necessario viene installato come dipendenza.

Il **touchpad** è un *Elantech*, ma il driver *elantech* è disponibile solo per kernel superiori al 2.6.28 (o come patch per kernel inferiori)... Poco male dato che viene comunque riconosciuto e funziona perfettamente con il driver *mouse* di xorg.

Entrando nei dettagli, possiamo contare su:

- scroll verticale, agendo con due dita sul touchpad;
- trascinamento, premendo con il dito e spostandosi senza rilasciare la pressione;
- emulazione del tasto centrale del mouse, mediante tocco con due dita.

A questo punto non ci resta che scrivere il file di configurazione xorg.conf. Con la versione *xorg 7.3* presente in Lenny, il file non sarebbe più necessario, ma scriverlo è comunque un buon esercizio per prendere pratica e conoscere meglio il sistema. Dato che ci siamo, possiamo abilitare lo schermo composito.

```
Section " InputDevice"
        Identifier       "Generic Keyboard"
        Driver           "kbd"
        Option     "XkbRules"    "xorg"
        Option     "XkbModel"    "pc105"
        Option     "XkbLayout"   "it"
EndSection

Section "InputDevice"

        Identifier      "Configured Mouse"
        Driver          "mouse"
EndSection

Section "Monitor"
       Identifier      "Integrato"

EndSection

Section "Screen"
        Identifier      "Default Screen"
        Device          "Intel Corporation Mobile 915GM /GMS/910GML Express
                                                        Graphics Controller"
        Monitor         "Integrato"
        DefaultDepth    24
        SubSection "Display"
                Depth           24
                Virtual         2048 2048
        EndSubSection
EndSection
```

```
Section "Device"
        Identifier      "Intel Corporation Mobile 915GM/GMS/910GML Express
                                                    Graphics Controller"
        Driver          "intel"
        BusID           "PCI:0:2:0"
        Option          "EnablePageFlip"    "on"
        Option          "AccelMethod"        "EXA"
        Option          "MigrationHeuristic"   "greedy"
        Option          "AccelDFS"      "true"
        Option          "DynamicClocks"    "on"
        Option          "ColorTiling"       "on"
        Option          "FBTexPercent"       "95"
        Option          "XAANoOffscreenPixmaps" "true"
        Option          "AddARGBGLXVisuals"     "true"
EndSection


Section "DRI"
    Mode 0666
Endsection


Section "Extensions"
    Option              "Composite"     "Enable"
Endsection
```

Le prestazioni della scheda video non sono niente male, un test con *glxgears* indica:

```
7063 frames in 5.0 seconds = 1412.485 FPS
7045 frames in 5.0 seconds = 1408.881 FPS
7063 frames in 5.0 seconds = 1412.452 FPS
7043 frames in 5.0 seconds = 1408.592 FPS
```

Attivando il compositor integrato in xfwm4 (Xfce) le prestazioni però calano un po'...

```
3179 frames in 5.0 seconds = 635.725 FPS
3195 frames in 5.0 seconds = 638.996 FPS
3125 frames in 5.0 seconds = 624.923 FPS
3158 frames in 5.0 seconds = 631.504 FPS
```

Desktop environment

A questo punto installiamo il nostro DE preferito. **Xfce** può essere un'ottima scelta, dato che offre un DE completo ma minimale e bisognoso di poche risorse sia in termine di RAM e CPU che di spazio su disco. Niente esclude che si possa installare qualunque altro DE. Se vogliamo optare per **Fluxbox**, prima leggiamo l'articolo dedicato, presente in questa rivista.

Per Xfce:

```
# aptitude install xfce4
```

Possiamo scegliere se installare o meno un gestore di login grafico. Possiamo scegliere tra slim, xdm o gdm. Se non lo vogliamo ma vogliamo comunque che Xfce venga avviato in automatico dopo il login testuale sulla *tty1* possiamo aggiungere a *∼/.bashrc* dell'utente:

```
if [ "`tty`" = "/dev/tty1" ]
then
  startxfce4 1>/tmp/startxfce.log 2>&1
fi
```

In questo modo registriamo anche un log in */tmp*. Adesso possiamo arricchire il nostro desktop con le nostre applicazioni preferite.
Ricordiamo di aggiungere il nostro utente al gruppo **powerdev** per poter dare l'halt al sistema sfruttando *hal* e *dbus*, passando per l'interfaccia grafica.

```
# adduser <nomeuser> powerdev
```

Audio

```
00:1b.0 Audio device: Intel Corporation 82801G (ICH7 Family) High Definition Audio
                                                              Controller (rev 02)

chip:  Realtek ALC269
```

Con la versione *alsa-base 1.0.17* disponibile in Lenny, il microfono integrato dà seri problemi di utilizzo. Per questo siamo costretti ad utilizzare una versione più aggiornata. Possiamo prelevare l'ultima versione dal sito ufficiale oppure più semplicemente, la preleviamo dai repository **experimental** di Debian, in cui è disponibile la versione *1.0.18*. Abilitiamo il repository aggiungendo al file *sources.list*:

```
deb ftp://ftp.debian.org/debian experimental main
```

Quindi:

```
# aptitude update
# aptitude install module-assistant
# m-a prepare
# aptitude -t experimental install alsa-source
# m-a build alsa
# m-a install alsa
```

Al riavvio avremo risolto i nostri problemi. L'*alsamixer* indica i seguenti canali:

```
PCM = canale delle casse integrate

LineOut = canale delle cuffie

iSpeaker = interruttore mute/unmute

Capture = volume del microfono

Digital = amplificazione del microfono
```

Dopo una serie di prove sul microfono, il giusto livello dei canali che garantiscono un ottima resa di registrazione senza fruscii fastidiosi e distorsioni è risultato essere:

```
Capture = 87
Digital = 37
```

Noto quanto sopra ci prepariamo ad abilitare i controlli audio con i tasti funzione sfruttando il sistema *acpi* (indipendente dal DE scelto):
Installiamo gli script per il supporto *acpi* dell'eeePC

```
# aptitude purge acpi-support
# aptitude install eeepc-acpi-scripts
```

In questo modo possiamo contare su ulteriori features quali, lo spegnimento del sistema in seguito alla pressione del tasto di accensione/spegnimento e l'ibernazione in seguito alla chiusura dello schermo che è perfettamente funzionante.
Quindi modifichiamo opportunamente il file */etc/default/eeepc-acpi-scripts* in modo che riporti al suo interno, tra le altre cose:

```
VOLUME_LABEL='PCM'
HEADPHONE_LABEL='iSpeaker'
I_SWITCH_LABEL='iSpeaker'
```

Così possiamo controllare i giusti canali audio con i tasti funzione.

Webcam

```
Bus 005 Device 004: ID 093a:2700 Pixart Imaging, Inc.
```

La webcam da 0.3 megapixel è perfettamente riconosciuta e supportata dal driver *uvcvideo* già incluso nel kernel 2.6.26. È necessario eventualmente caricare il modulo *uvcvideo* e attivare la webcam con:

```
# echo "1" > /sys/devices/platform/eeepc/camera
```

Si può configurare e verificarne il corretto funzionamento mediante il programma *luvcview* avviabile dalla shell dell'utente con:

```
$ luvcview -f yuv
```

Batteria

La batteria in dotazione ha 4 celle per una capacità totale di 4,4 AH che consentono una durata media in condizioni operative normali di 4 ore circa. Xfce ne permette il monitoraggio e la gestione grazie all'applet **xfce4-battery-plugin** posizionabile sul pannello.

Processore: hyper-threading e scaling

A differenza degli altri modelli della stessa serie che montano un Celeron™, il 900A è equipaggiato con un processore **Intel® Atom™ N270 Diamondville™** a 1,6 GHz con bus a 533 MHz, architettura X86 (32 bit) senza implementazione EM64T.
Alimentato con una tensione di 1.2 V, questo processore dissipa solamente 2.5W, ideale per una lunga durata della batteria. Altre caratteristiche sono il supporto alla tecnologia hyper-threading e cache di L1=32KB e L2=512 KB.
Il modulo del kernel *acpi_cpufreq* si occupa dello scaling del processore e viene caricato e attivato in automatico senza intervento alcuno. Ogni personalizzazione con il nostro strumento preferito *(cpufrequtils* è quello già installato di default) è ovviamente possibile.

Come possiamo notare da */proc/cpuinfo* il processore supporta due step, 800 e 1600 MHz. Notiamo che il kernel vede due processori, anziché uno, dimostrazione del fatto che l'hyper-threading è attivo e supportato dal kernel.
Da dmesg:

```
...
[    0.243959] CPU1: Intel(R) Atom(TM) CPU N270   @ 1.60GHz stepping 02
[    0.243997] checking TSC synchronization [CPU\#0 -> CPU\#1]: passed.
[    0.248015] Brought up 2 CPUs
[    0.248015] Total of 2 processors activated (6388.74 BogoMIPS)
...
```

Dispositivo di archiviazione dati

Il modello monta un **SSD** (*solid state disk*) da 8 GB. Il disco a stato solido presenta principalmente il vantaggio di non avere meccanica in movimento, con conseguente maggiore silenziosità di utilizzo, maggiore risparmio energetico e resistenza agli urti. Le ipotesi mosse contro questo tipo di tecnologia di avere un ciclo di vita piuttosto limitato sembrano essere recentemente smentite [3] e la vita di un SSD sottoposto ad un utilizzo normale si estende per diversi anni.

Un test con *hdparm* sul disco a stato solido restituisce le seguenti caratteristicge (valore medio di 10 repliche)

```
/dev/sda:
 Timing cached reads:   1208 MB in  2.00 seconds = 603.78 MB/sec
 Timing buffered disk reads:   86 MB in  3.07 seconds =  28.05 MB/sec
```

[3] http://wiki.eeeuser.com/ssd_write_limit

Possiamo comunque limitare al massimo la frequenza di scrittura su disco, con un guadagno anche in prestazioni, aggiungendo ad "/etc/sysctl.conf" la seguente opzione:

```
vm.dirty_writeback_centisecs = 1500
```

In questo modo impostiamo che i dati su disco (es. log) vengano scritti ogni 15 secondi anziché ogni 5.

Possiamo montare le directory

```
/tmp
```

```
/var/run
```

```
/var/lock
```

su fs temporaneo in RAM. L'ulteriore vantaggio risiede nel fatto che la scrittura/lettura su RAM è più veloce, ma ovviamente tutti i dati in queste directory verranno inevitabilmente persi ad ogni reboot. Si aggiunge ad */etc/fstab* la seguente stringa:

```
tmpfs /tmp tmpfs defaults 0 0
```

Poi si modifica il file */etc/default/rcS* in modo che riporti:

```
...
RAMRUN=yes
RAMLOCK=yes
...
```

Boot più snello

Valgono le stesse raccomandazioni per ogni altro sistema: non avviare i servizi e demoni che non sono necessari.
In particolare si può disabilitare lo script */sbin/hwclock* con:

```
# chmod -x /sbin/hwclock
```

Questo riduce il tempo di boot di molti secondi (circa 10) portandolo a circa 20 s, senza nessun effetto collaterale riscontrato.
Si può utilizzare **dash** (la *Debian Almquist Shell*) invece di *bash* in fase di boot, snellendo ulteriormente i tempi:

```
# aptitude install dash
# dpkg-reconfigure dash
```

La shell utilizzata dai vari utenti comunque non cambia dato che è impostata in */etc/passw*.

```
Happy Debian, happy hacking ;-)
```

Capitolo 6

Tips & Tricks

Che aggiungere, soffiate e trucchi ;-)
Tutto ciò di utile che non rientra nelle altre categorie.
In questa sezione troverete articoli che riguardano questioni settoriali e/o molto specifiche su di un argomento in particolare.

6.1 Crittografia portami via

Come proteggere i nostri dati dagli intrusi? La risposta è semplice ed è **crittografia**. Su linux in generale è molto facile ottenere ciò, su Debian e derivate ancora di più. Tutto quello che seguirà è stato testato sia su una Debian Etch che su una Debian Lenny.
Non faremo altro che creare una directory crittografata usando alcuni semplici strumenti forniti nei repository di debian.

6.1.1 Installazione

Ci servono sostanzialmente queste due cose:

- fuse: in sostanza permette di creare dei filesystem virtuali, quindi non c'è bisogno di creare una partizione apposita per i nostri dati.
- encfs: crittografa un file system fuse con tre metodi, uno più sicuro dell'altro.

Quindi ora installiamo questi due componenti che si porteranno dietro anche le dipendenze necessarie.

```
# apt-get install fuse-utils encfs
```

Fuse è un modulo del kernel quindi bisogna installarlo tramite module-assistant, nel caso non abbiate questo tool

```
# apt-get install module-assistant build-essential
```

Da root lanciamo in successione:

```
# depmod -a
# modprobe fuse
```

Nota: su Debian Etch bisogna usare **module-assistant** dato che il sorgente del modulo è pacchettizzato singolarmente. Module-assistant permette di scaricare, compilare e pacchettizzare un modulo del kernel.
Lanciare adesso il comando

```
# m-a
```

e selezionare dal menù **PREPARE**: questo comando installerà i componenti essenziali per compilare moduli del kernel.
Fatto questo selezionare **SELECT** e scegliere *fuse* come modulo da installare. Procediamo con OK e, dal menù successivo, selezioniamo prima **GET** per scaricare i sorgenti del modulo fuse, poi **BUILD**.
Fatto questo i seguenti comandi servono: *depmod* genera un file di dipendenze tra i moduli, che poi viene utilizzato da *modprobe* per caricarli rispettando le dipendenze. Precisamente, viene creato il file */lib/modules/* versione */modules.dep*. Modprobe invece carica un modulo "al volo". [a]

[a] per ulteriori informazioni: http://paper0k.wordpress.com/2007/02/20/i-moduli-del-kernel/

Inoltre dobbiamo modificare il seguente file di configurazione, in modo che il modulo fuse venga caricato all'avvio del sistemas Sempre da root , con il nostro editor di testo preferito apriamo il file:

```
/etc/modules
```

All'interno del file dobbiamo aggiungere la riga `fuse`, quindi salvare e chiudere il file.
Ora è necessario riavviare il computer per vedere se tutto è andato per il verso giusto.

6.1.2 Utilizzo

Ok il più è fatto, ora abbiamo tutto il necessario per aggiungerci al gruppo fuse. In questo modo possiamo crearci un file system nella nostra home o dove ci pare (basta che abbiamo i permessi adatti in quella partizione o directory o che ne so io...):

```
# adduser tuonick fuse
```

Abbiamo due scelte da seguire: usare il terminale, oppure utilizzare un programmino in python pensato per KDE.

Via terminale

Avviate il terminale e per creare le cartelle che vi servono digitate (senza i permessi di root, la cartella è vostra e root non c'entra più nulla, neanche lui potrà accedere ai vostri dati, né la NASA, né la CIA, né la vostra ragazza!)

```
encfs ~/.crypto_crypt ~/crypto

La directory "/home/ettore/.crypto_crypt/" non esiste. Deve essere creata? (s, n) y
La directory "/home/ettore/crypto" non esiste. Deve essere creata? (s, n) y
Creazione nuovo volume cifrato.
```

Ovviamente con i nomi che volete voi.
Fate attenzione che il primo nome *.crypto_crypt* sarà la directory nascosta con i dati criptati.
Il secondo nome *crypto* sarà la directory che una volta montata conterrà i nostri dati da criptare
Dato che il comando è stato eseguito da */home/user* le directory saranno */home/user/.crypto_crypt* e */home/user/crypto* e la seconda directory *crypto* sarà visibile anche al browser.
Ora occorre scegliere la configurazione di criptazione:

```
Scegliere tra una delle seguenti opzioni:
 inserire "x" per la modalità di configurazione per esperti,
 inserire "p" per la modalità paranoica preconfigurata.
 qualsiasi altra cosa o una riga vuota selezionerà la modalità standard.
?>
.
Selezionata la configurazione standard.
.
Configurazione terminata. Il filesystem verrà creato con
le seguenti proprietà:
Cifrario file system: "ssl/aes", versione 2:1:1
Codifica nome dei file: "nameio/block", versione 3:0:1
Dimensione della chiave: 192 bits
Grandezza blocco: 1024 byte
Ogni file contiene 8 byte di intestazione con dati "unique IV".
Nomi dei file cifrati usando la modalità di concatenazione IV ("IV chaining").
```

Qui a meno che non abbiate preferenze particolari , premete invio per la configurazione standard.

Adesso passiamo all'ultima fase che è la scelta della password:

```
Ora è necessario creare una password per il proprio file system.
È necessario ricordare questa password, dato che non esiste alcun modo
per recuperarla automaticamente. In ogni caso , la password
potrà essere cambiata utilizzando encfsctl.
Nuova password di Encfs:
Verifica password di Encfs:
```

Con questo metodo inizialmente il sistema crea e monta le directory, successivamente (con le directory già esistenti) lo stesso comando monterà il filesystem virtuale chiedendovi la password inserita. Ora sulla directory nella vostra home ci potete navigare anche a finestra o via terminale, come preferite.
Una volta finite le vostre operazioni smontate la directory con

```
fusermount -u crypto
```

Vi accorgerete navigando che la directory */home/user/crypto* sarà vuota mentre */home/user/.crypto_crypt* conterrà i dati criptati illeggibili.
Se volete riaccedervi dovete dare il comando iniziale di creazione:

```
encfs ~/.crypto_crypt ~/crypto
```

E inserire quando richiesto la password generata alla creazione delle cartelle criptate
Se volete eliminare tutto , vi basta cancellare */home/user/crypto* e */home/user/.crypto_crypt* e non resterà traccia delle partzioni criptate

Per comodità vi potete creare i comandi con alias in */home/user/.bashrc* in modo che l'utente non debba scrivere tute le volte i comandi sopracitati, ad esempio :

```
alias crypton='encfs ~/.crypto_crypt ~/crypto'
alias cryptof='fusermount -u crypto'
```

in questo modo , come user se scrivo *crypton* monto il filesystem criptato e con *cryptof* la smonto.
In questo caso i nomi del comando di mountaggio e smontaggio sono a vostra scelta

Per via grafica

Si può usare un programmino in python per KDE: **K-EncFS**
Naturalmente su linux ci sono miliardi di modi per fare la stessa cosa. In ambiente desktop di uso quotidiano l'uso precedente del terminale può diventare noioso. Quindi, se siamo utenti che usano il DE KDE andate qui [1] e scaricare il il file compresso contenente K-EncFS. Dentro al tar.bz2, c'è la versione .deb per Debian che ovviamente, prima di installare dovrete scopattare con ark o con tar nella vostra home. Prima di procedere abbiamo bisogno di soddisfare tutte le dipendenze per questo programma:

```
# apt-get install python python-qt3 python-kde3
```

Ora non dovete fare altro che installare K-EncFS con:

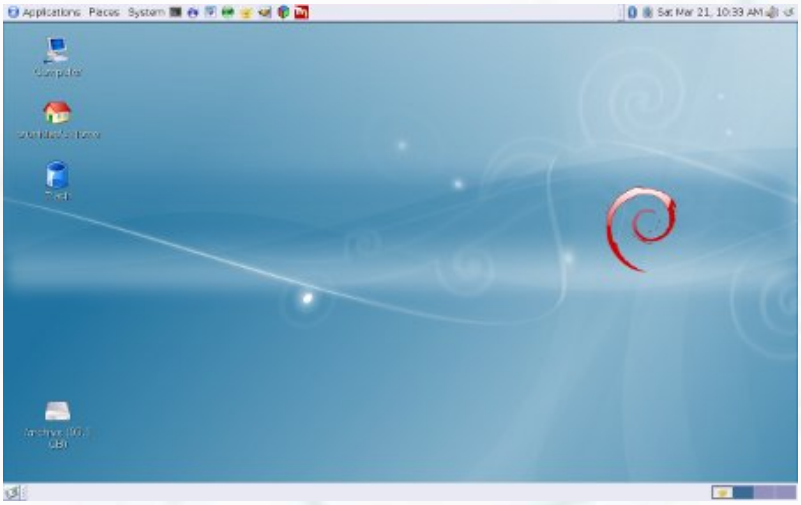
```
# dpkg -i kencfs2_2.1-1trisz_all.deb
```

Il programmino (*k menù* → *accessori*) è molto intuitivo, monta e smonta con password il filesistem criptato in una directory nascosta predefinita nella vostra home. Potete anche scegliere qui, che metodo usare per la crittografia ma basta quello predefinito, dato che è quello ufficialmente usato e riconosciuto dalla NASA ;-).
Fate attenzione perchè con l'interfaccia grafica di *kencfs2* non sarete in grado di riconoscere e montare il filesystem criptato creato da terminale , ma dovrete creare un nuovo filesystem criptato che sarà in */home/user/.kencfs2/encrypted* e con la gui potrete sapendo la password montarla e smontarla. Sono **due metodi differrenti** che potete usare entrambi, ma ricordate che sono 2 cose indipendenti e separate.

[1] http://www.kde-apps.org/content/show.php/K-EncFS?content=54078

Capitolo 7

Interfacce grafiche



Analisi e approfondimenti sulle interfacce grafiche.
Soprattutto desktop, ma non solo.

In questa categoria presenteremo tuning e configurazioni delle varie interfacce grafiche possibili e compatibili con la nostra debian.

7.1 Fluxbox: Installazione e configurazione

L'installazione del window manager fluxbox è banale:

```
# apt-get install fluxbox
```

Il mio gestore di login è *kdm*, per cui l'integrazione è stata immediata ed automatica: al successivo riavvio è stato possibile scegliere la sessione come *fluxbox* anziché kde.
Tutto sembra funzionare, se non che tutte le modifiche a *∼/.fluxbox/startup* sono ignorate.

7.1.1 Avvio di fluxbox

L'avvio di fluxbox dal gestore di login kdm avviene andando a leggere il file **/usr/share/xsessions/fluxbox** il cui contenuto è il seguente:

```
[Desktop Entry]
Encoding=UTF-8
Name=Fluxbox
Comment=Highly configurable and low resource X11 Window manager
Exec=/usr/bin/startfluxbox
Terminal=False
TryExec=/usr/bin/startfluxbox
Type=Application

[Window Manager]
SessionManaged=true
```

Se le linee che incominciano con `Exec` e `TryExec` contengono solo fluxbox, è necessario modificarle come sopra (sostituendo fluxbox con /usr/bin/startfluxbox).
In questo modo si carica la sessione fluxbox non con la semplice chiamata all'eseguibile fluxbox, ma utilizzando l'apposito script **/usr/bin/startfluxbox**.

Lo script /usr/bin/startfluxbox

Questo serve a lanciare fluxbox con il profilo utente in uso e quindi di utilizzarne i file di configurazione, altrimenti ne vengono creati di default. Il codice è il seguente:

```
#!/bin/sh

command="`basename \"$0\"`"
fluxdir="$HOME/.fluxbox"
startup="$fluxdir/startup"

while [ $# -gt 0 ]; do
    case "$1" in
        -c|--config)
            if [ $# -lt 2 ]; then
                echo "$command:error, missing argument"
                exit 1
            fi
            shift
            startup=$1
        ;;
        -h|--help) cat <<< EOF
#!/bin/sh
#
# fluxbox startup-script:
#
# Lines starting with a '#' are ignored.

# Change your keymap:
xmodmap "$HOME/.Xmodmap"

# Applications you want to run with fluxbox.
# MAKE SURE THAT APPS THAT KEEP RUNNING HAVE AN ''&'' AT THE END.
#
# unclutter -idle 2 &
# wmnd &
# wmsmixer -w &
# idesk &

# And last but not least we start fluxbox.
# Because it is the last app you have to run it with ''exec'' before it.

exec fluxbox
```

```
# or if you want to keep a log:
# exec fluxbox -log "$fluxdir/log"
EOF
    ) > "$startup"
    fi
    chmod 644 "$startup"
    exec sh "$startup"
fi
```

questo non fa altro che lanciare il file *∼/.fluxbox/startup*. Se tale file non esiste verrà riavviato l'X server e ci si ritroverà di nuovo al kdm.

lo script ∼/.fluxbox/startup

Uso questo script per lanciare l'eseguibile fluxbox vero e proprio e fare altre cosette...
Questo è il codice di *∼/.fluxbox/startup*:

```
# Applications you want to run with fluxbox.
# MAKE SURE THAT APPS THAT KEEP RUNNING HAVE AN & AT THE END.
#
# unclutter -idle 2 &
# wmnd &
# wmsmixer -w &
idesk &

# And last but not least we start fluxbox.
# Because it is the last app you have to run it with exec before it.

exec /usr/bin/fluxbox
# or if you want to keep a log:
# exec /usr/bin/fluxbox -log "/home/ettore/.fluxbox/log"
```

Toglieremo il "cancelletto" (#) ad idesk (come mostrato è il risultato finale) per configurare al meglio un file manager. Nel prossimo capitolo vi spiegheremo come fare.

7.1.2 Prima configurazione

Installare un file manager

Per visualizzare un desktop a finestre tradizionale è necessario installare idesk (per creare delle icone sul desktop, sia per avviare il file manager, sia per avviare qualsiasi altra applicazione) e un file manager. Data la leggerezza e la scarsità di dipendenze consigliamo l'utilizzo di *pcmanfm*. Incominceremo dunque ad installare i seguenti pacchetti:

```
# aptitude install idesk pcmanfm
```

idesk

Se l'installazione del file manager non richiede particolari attenzioni, idesk deve essere completamente configurato. Istruzioni per il suo utilizzo le troveremo nel file */usr/share/idesk/README*. Innanzi tutto creeremo la directory *.idesktop* nella nostra home:

```
~$ mkdir .idesktop
```

All'interno di questa directory andremo a creare un file *nomechevogliamo.lnk* che corrisponderà ad un'icona sul desktop. Se il nome di questi files è irrilevante, vi ricordo che la denominazione *.lnk* è obbligatoria. Nel file appena creato daremo dei parametri per lanciare l'applicazione voluta. Qui sotto un esempio di un'icona per avviare il file manager nella nostra directory d'utente principale:

```
table Icon

Caption: My_home
ToolTip.Caption: La mia Home
Command: pcmanfm /home/$USER
Icon: /usr/share/idesk/folder_home.xpm
Width: 48
Height: 48
X: 59
Y: 129
end
```

Dove `Caption` è il nome sottostante all'icona, `ToolTip.Caption` la descrizione che apparirà al passaggio del mouse sopra l'icona, `Command` è il comando che viene dato al doppio click dell'icona (in questo caso lanciamo il file manager nella directory principale dell'utente), `Icon` è l'icona che appare a video, `Width` e `Height` sono le dimensioni dell'icona ed infine `X` e `Y` sono la posizione dell'icona all'avvio del nostro window manager rispetto al desktop. Se abbiamo capito il sistema, qui sotto un altro esempio di un'icona che avvia *iceweasel*.

```
table Icon
Caption: Iceweasel
ToolTip.Caption: Web Browser Iceweasel
Command: iceweasel
Icon: /usr/share/iceweasel/icons/mozicon50.xpm
Width: 48
Height: 48
X: 59
Y: 219
end
```

Inoltre nel file *∼/.ideskrc* troveremo altre opzioni possibili da attribuire alle icone (come font della scritta, dimensione, background, ...).

Immagine di Background

Dopo il primo avvio di fluxbox, un messaggio di errore ci informerà che *fbsetbg* non trova un programma per gestire il background. Nella finestra d'informazione ci viene consigliata l'installazione di *eterm*, il quale contiene *esetroot* che ci consentirà di gestire/installare uno sfondo. Installiamo dunque il programma:

```
# apt-get install eterm
```

Per controllare che tutto sia andato per il meglio, così come per controllare la situazione se non abbiamo ricevuto nessun messaggio d'errore all'avvio, digitiamo il comando:

```
$ fbsetbg -i
```

il quale dovrebbe darci un output tipo:

```
Esetroot is a nice wallpapersetter. You won't have any problems.
```

È possibile utilizzare anche altri programmi allo stesso scopo (come ad esempio feh); l'importante è che quest'ultimo comando ci dia il rassicurante You won't have any problems. Per selezionare ora un background daremo il comando:

```
$ fbsetbg <path_immagine_sfondo>
   (es. fbsetbg /home/$USER/Pictures/background.jpg)
```

Verrà a questo punto inserito lo sfondo voluto e creato un file lastwallpaper nella directory *∼/.fluxbox* che conterrà il path di quest'immagine. Di default, al prossimo avvio verrà ripescata l'immagine di sfondo grazie proprio a questo file. Il pacchetto fbsetbg permette anche altre configurazioni (come ad esempio una funzione random di un'immagine a caso da una directory, ...). Per le opzioni possibili si consulti il manuale .

```
$ man fbsetbg
```

Il file ∼/.fluxbox/menu

È il file di configurazione del menu fluxbox. Ciascun utente può personalizzarlo liberamente nella propria home
Di default il file contiene un include al file */etc/X11/fluxbox/fluxbox-menu*, il quale a sua volta contiene la configurazione del menu di default. Per scrivere un proprio menu si può prendere spunto dal file antecedentemente citato o in modo più semplice, si può partire dal seguente template:

```
[begin](Fluxbox-0.9.12)
[exec] (Shell) (xterm)
[exec] (Browser) {firefox}
[end]
```

dove tra parentesi () troviamo il nome che comparirà a video e tra parentesi graffe {} inseriremo il nome del comando che avvierà l'applicazione. Per inserire un sotto-menu è necessario nidificare:

```
[submenu] (nome_del_sotto-menu)
[exec] (nome_a_video) {comando_avvio}
[end]
```

Un sotto-menu può avere a sua volta altri sotto-menu nidificati ricorsivamente.

Icone nei menu

Per associare un'icona ad una voce di menu va seguita la sintassi:

```
[submenu] (nome_del_sotto-menu)
[exec] (nome_a_video) {comando_avvio} <file_icona_in_formato.xpm>
[end]
```

Slit

Trovate la documentazione ufficiale a questo indirizzo.[1]
La slit è una barra in cui possono essere raccolte applicazioni *dockable*. Per precisazione, NON si tratta della toolbar di fluxbox, ma di una barra a parte. Solitamente le applicazioni dock vanno lanciate con l'opzione `-w` in moto tale che non creino una propria finestra sul desktop, ma vengano raccolte una sull'altra.
Vanno normalmente avviate con lo script di inizializzazione del window manager (nel notro caso *~/.fluxtbox/startup*) e poi ordinate secondo l'ordine di *~/.fluxtbox/slitlist*, o comunque il file specificato come slitlist in *~/.fluxtbox/init*. Per fare un esempio, fra queste applicazioni troviamo *wmacpi* (marca lo stato della batteria dei laptop), *wmcpu* (informazioni sulla cpu), *wmcalc* (calcolatrice), ...
Nel prossimo articolo troviamo un esempio di configurazione di fluxbox sull'eeePC.

[1] http://fluxbox.sourceforge.net/docbook/it/html/chap-slit.html

7.2 Configurazione su eeePC

Visto il notevole successo riscosso dai netbook, i portatili di piccola taglia, ho pensato di riportare in questa piccola guida tutta la serie di modifiche che ho effettuato sul mio eeepc per ottimizzarne l'utilizzo con fluxbox.
Prima di cominciare, se avete anche voi un eeepc vi consiglio di leggere attentamente il capitolo 5 di questa rivista, per configurare e risolvere i principali problemi con debian sul vostro portatile.
La guida è stata testata personalmente su un eeepc 900 (display 9) ma è adatta a qualsiasi portatile con display di dimensioni ridotte.

7.2.1 Scorciatoie da tastiera

Uno degli strumenti più potenti che ci mette a disposizione fluxbox sono le scorciatoie da tastiera: tramite il file *~/.fluxbox/keys* possiamo intercettare combinazioni di tasti (anche multiple, come in *emacs*) ed assegnare loro una azione. L'azione può essere un comando interno di fluxbox o un comando della shell e nelle ultime versioni possiamo anche intercettare i click del mouse. Un elenco completo ed aggiornato dei comandi utilizzabili lo trovate sul wiki ufficiale [2].

Controllo dei workspace

Alcune di queste scorciatoie sono già impostate nella configurazione di default di fluxbox, quindi non copia-incollate tutto ma controllate di non inserire righe duplicate.
Nota: di default fluxbox imposta il cambio di workspace con *Ctrl+Fx*, mentre io preferisco utilizzare *Alt+Fx*. Cambiate le impostazioni in base alla combinazione con cui vi sentite a vostro agio.

```
Mod1 Tab :NextWindow
Mod1 Shift Tab :PrevWindow
Mod1 F1 :Workspace 1
Mod1 F2 :Workspace 2
Mod1 F3 :Workspace 3
Mod1 F4 :Workspace 4
Control Mod1 Left :LeftWorkspace
Control Mod1 Right :RightWorkspace
```

[2] http://fluxbox-wiki.org/index.php?title=Keyboard_shortcuts#Fluxbox_Keycommands

Nota: *Mod1* corrisponde al tasto Alt. In questo modo ci potremo spostare tra le finestre con il classico Alt+Tab, andare sul workspace X con Alt+FX e potremo spostarci a sinistra/destra con Ctrl+Alt+freccia come in GNOME.

Controllo delle finestre

Come tutti i portatili eeepc è dotato di touchpad per il puntamento, quindi è scomodo dover ricorrere al click del mouse per compiere delle azioni sulla finestra come spostarla di workspace, ingrandirla, metterla in primo piano, chiuderla. Ecco qualche combinazione che fa al caso nostro:

```
Mod4 t :ToggleDecor
Mod4 i :Maximize
Mod4 c :Close
Mod4 k :KillWindow
Mod4 s :Stick
Mod4 r :Raise
Mod4 l :Lower
```

Nota: *Mod4* corrisponde al tasto windows, che finalmente nell'eeepc è sostituito da una casa.

- *ToggleDecor* toglie le decorazioni dalla finestra (barra del titolo, bordi). Utile se volete aumentare lo spazio a disposizione della vostra finestra.
- *KillWindow* è comodissimo: chiude una finestra killando il processo che la tiene aperta. Equivale ad un *xkill+click* sulla finestra. Quando si impalla qualche programma beta premete un tasto e il problema sparisce.

Come sempre vi consiglio di fare riferimento alla documentazione ufficiale per avere un elenco di ulteriori azioni che possono essere eseguite.

Avvio di programmi esterni

Dato che ci sono una serie di programmi che abbiamo bisogno di avviare spesso può risultare scomodo tutte le volte attraversare il folto menu di debian per trovarli, specialmente con il touchpad che rende difficile spostare il puntatore nel sotto-menu giusto.

Impostiamoci allora alcune scorciatoie per eseguire velocemente i programmi di uso più frequente:

```
Print :ExecCommand import -window root $HOME/screen-‘date +%d%m%y-%H%M%S‘.png
Control Print :ExecCommand import -window 0 $HOME/screen-‘date +%d%m%y-%H%M%S‘.png
Mod1 Print :ExecCommand import $HOME/screen-‘date +%d%m%y-%H%M%S‘.png

Control Mod1 p :ExecCommand gcolor2
Control Mod1 t :ExecCommand xterm
Control Mod1 b :ExecCommand opera
Control Mod1 r :ExecCommand fbrun
```

Le prime tre sono per effettuare screenshots con imagemagick; poi abbiamo una scorciatoia per il color picker, una per il browser, una per il terminale ed una per fbrun (l'esegui comando di fluxbox); voi siete liberi di crearne quante volete!

7.2.2 Menu personalizzato

Rimandiamo all'articolo precedente, sezione 7.1.2. Per approfondimenti vi rimandiamo alla documentazione ufficiale.

Applicazioni esterne

Ora pensiamo a rendere fluxbox più usabile tramite alcune applicazioni esterne. Comincio da quelle più strettamente utili fino a terminare con quelle mirate a migiliorare l'aspetto del nostro desktop: anche l'occhio vuole la sua parte.

fbpager: un pager per fluxbox

Grazie al pager possiamo capire con un solo colpo d'occhio su quale workspace ci troviamo e dove sono le nostre finestre: installiamolo dal repository con

```
# apt-get install fbpager
```

...e poi andate a modificare il suo file di configurazione *∼/.fluxbox/fbpager* per renderlo grande quanto la toolbar:

```
fbpager.alpha: 200
fbpager.x: 0
fbpager.y: 0
```

```
fbpager.workspace.width: 20
fbpager.workspace.height: 20
```

dockapps: teniamo sotto controllo il sistema

Molti di noi debianizzati, specialmente se novizi, sono abituati alle applet dei pannelli dei DE e si trovano spaesati davanti ad un wm privo di strumenti che mostrino lo stato del sistema. Ci vengono inconto le dockapps, piccole applicazioni studiate per windowmaker ma che si integrano perfettamente in fluxbox, grazie allo slit. Lo slit è un vassoio invisibile che ospita questo tipo di applicazioni e ci permette di posizionarle nel bordo dello schermo preferito.
Molte dockapps si trovano già nei repository, se volete l'elenco completo date il comando

```
$ apt-cache search ^wm
```

...e leggendo la descrizione decidete se installarle ed eseguirle. Le dockapps che utilizzo sono:

- **wmacpi**, per tenere d'occhio la carica della batteria;
- **wmmon**, per monitorare il carico della cpu e notare se ci sono processi impazziti;
- **wmweather**, per leggere il bollettino meteo;
- **wmmixer**, per vedere il volume del mixer.

Ne trovate per tutti i gusti.

parcellite: un gestore di appunti

Sotto kde ero diventato a tutti gli effetti *klipper*-dipendente: una volta passato a fluxbox sentivo la mancanza di uno strumento che memorizzasse tutti i miei *ctrl+c* e mi consentisse di richiamare gli appunti copiati precedentemente. Purtroppo sia *klipper* che *glipper* (la variante gnome) richiedono le librerie dei loro rispettivi ambienti desktop e non mi sembrava il caso di sporcare il sistema in questo modo. Poi ho scoperto parcellite, che fa lo stesso lavoro ed è scritto in gtk+ :)
Se anche voi siete *g/klipper*-dipendenti installatelo e vi sentirete a casa; lo trovate nei repository ufficiali.

wbar: una dockbar in stile osx

Non sono certo un apple fanboy ma devo ammettere che questa dockbar ha il suo fascino e può tornare utile per tenere a portata di mano alcune applicazioni. Non è ancora presente nei repository quindi dovete compilarvela a manina. È bene avere un minimo di conoscenza nella compilazione di programmi, quindi se non sapete come farlo è la volta buona per imparare.

```
$ wget http://freshmeat.net/redir/wbar/66660/url_mirror/wbar-1.3.3.tbz2
$ tar -jxvf wbar-1.3.3.tbz2
$ cd wbar-1.3.3/
$ make
# checkinstall
```

Nota: durante la compilazione potreste ottenere un errore del tipo: IconLoader.cc:20: error: 'getenv' was not declared in this scope per ripararlo, aggiungete la seguente riga:

```
#include <cstdlib>
```

all'inizio dei file: `cIonLoader.h` e `SuperBar.h`

setbg-rc: uno script personale per impostare lo sfondo scala-e-taglia

Fra le modalità supportate da *fbsetbg* per impostare lo sfondo manca quella che kde chiamava *scala-e-taglia*, ovvero una modalità che consente di mostrare l'immaggine in modo che occupi tutto lo schermo ma mantenendo le proporzioni originali (ovviamente a scapito di un ritaglio ai bordi dell'immagine). Per sopperire a questa lieve mancanza ho scritto uno script che lavora su una copia temporanea dell'immagine passata come argomento, la scala e la taglia usando imagemagick, la copia in *∼/.fluxbox/background.jpg* e la imposta come sfondo. È un po' lento, specialmente con foto ad alta definizione, ma non penso cambiate sfondo così spesso da non poter aspettare quattro-cinque secondi :D
Se vi interessa, copio ed incollo qua il codice:

```
#!/bin/bash

A=$RANDOM
mkdir -p /tmp/setbg
cp $1 /tmp/setbg/$A.jpg

cd /tmp/setbg
mogrify -resize x600 $A.jpg
convert -gravity Center -crop 1024x600+0+0  +profile \"*\" $A.jpg $A-ok.jpg

cp $A-ok.jpg ~/.fluxbox/background.jpg
cd
rm -r /tmp/setbg

fbsetbg .fluxbox/background.jpg
```

7.2.3 Conclusioni

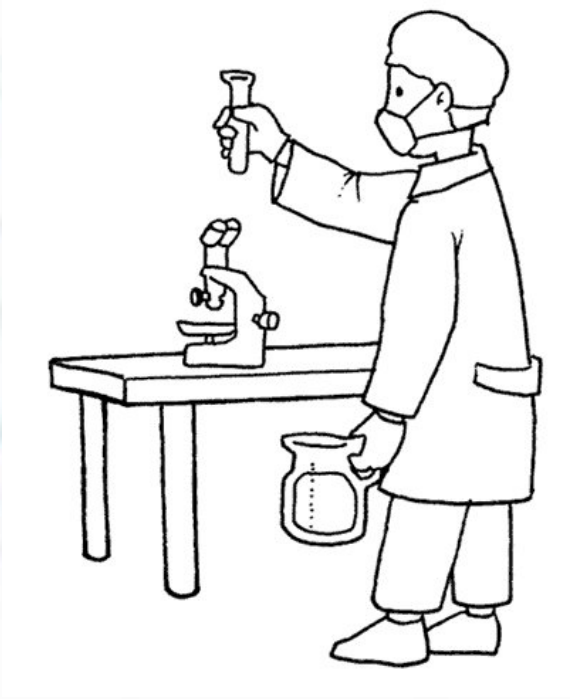
La guida è terminata, ora godetevi il vostro desktop rinnovato! In caso vi venissero altre idee per migliorare l'uso di fluxbox sui vostri netbook non esitate a contattare l'autore.

References:

- Fluxbox Homepage: http://www.fluxbox.org/version-0.9.php
- Homepage su sourceforge: http://fluxbox.sourceforge.net/
- Official fluxbox wiki: http://fluxbox-wiki.org/index.php
- A month with fluxbox: http://www.tuxmachines.org/node/392

Capitolo 8

Software in analisi



Approfondimenti e test su tutto il softwares per la nostra debian.
Il mondo del software libero ha raggiunto dimensioni più che ammirevoli.
In questa sezione cercheremo di presentarvi delle applicazioni utilizzabili con la nostra debian, nel modo più esaustivo possibile.

A seguire un articolo su uno degli editor di testo più conosciuti nell'ambiente unix: `vim`.

8.1 Guida a vim

8.1.1 Cos'è vim?

L'editor vim fornito con la maggior parte dei sistemi Linux è una versione estesa e migliorata dell'editor vi. vim include tutti i comandi e le funzionalità di vi, l'editor di UNIX, che rimane uno degli editor più utilizzati anche in sistemi Linux. La differenza tra un normale editor di testo (gedit per GNOME, kwrite per KDE) e vim è che quest'ultimo permette l'esecuzione di svariati comandi e la possibilità di creare comandi personalizzati, utilizzare script, gestire finestre di testo multiple. Inoltre aggiunge l'assistenza fornita ai programmatori grazie all'opzione di indentazione (le varie spaziature che si inseriscono all'interno di un sorgente per renderlo più leggibile) automatica e alla presenza di schemi di colori che rendono i codici più leggibili.
vim è l'editor di testo pensato per gli irriducibili della shell, e per tutti gli utenti che odiano spostare le mani dalla tastiera, quindi anche se al principiante potrebbe dare all'inizio l'impressione di essere un editor complesso, consiglio vivamente di provare almeno a imparare i comandi basilari in quanto è uno strumento indispensabile per la programmazione. Di seguito verranno analizzati quasi tutti i comandi utilizzati durante l'editing, spero che questa guida sia utile quanto lo è stata per me.

8.1.2 Iniziamo

In questo primo paragrafo descriverò come creare file di testo e cercare files già esistenti. Per lanciare vim apriamo la shell e digitiamo il comando:

```
christian@linux:~$ vim
```

e premiamo invio.
A questo punto saremo entrati in vim: il comando sopra esegue il programma. Se proverete a scrivere qualcosa vi accorgerete che in realtà non state scrivendo niente, non preoccupatevi è tutto normale, questo avviene perchè vim ha due modalità di funzionamento principali: "comando" e "inserimento". Quando lanciamo il programma questo parte in modalità comando, ovvero quella modalità in cui i tasti fanno parte delle opzioni disponibili, se vogliamo scrivere qualcosa dobbiamo passare in modalità inserimento con il seguente comando:

```
i
```

Adesso possiamo scrivere tutto quello che vogliamo come se fossimo in un normalissimo editor di testo. Dopo aver scritto tutto quello che vogliamo è giunto il momento di salvare il nostro lavoro, per fare questo dobbiamo tornare in modalità comando digitando:

```
esc
```

Adesso possiamo accedere a una terza modalità di `vim` detta "modalità ultima linea" digitando

```
:
```

Adesso vedrete apparire il simbolo ":" all'inizio dell'ultima riga dell'editor, e il cursore posizionarsi dopo di esso.Questa modalità di `vim` può essere considerata come una sorta di shell poichè dopo aver dato un comando per farglielo eseguire dovremo digitare il tasto invio. Adesso possiamo salvare il nostro file digitando:

```
:w nome_file
```

oppure

```
:sav nome_file
```

In entrambi i casi viene creato un file con il nome da voi scelto e salvato nella vostra home. Se dopo aver salvato il file volete uscire da `vim` digitate il comando

```
:q
```

e premete invio, in questo modo tornerete a visualizzare la schermata iniziale della vostra shell. Un buon sistema per velocizzare il tutto è quello di salvare il file e uscire da `vim` digitando i due comandi precedentemente descritti in un'unica azione:

```
:wq nome_file
```

Quando poi abbiamo bisogno di recuperare il file salvato non facciamo altro che digitare da terminale:

```
christian@linux:~$ vim nome_file
```

e il file verrà aperto, ricordate che quando dovete modificare file di sistema o file critici dovete loggarvi come root altrimenti vi dirà che non avete i permessi di scrittura sul file. A questo punto una volta aperto il file e fatto le dovute modifiche torniamo in modalità comando e digitiamo " `:w` " per salvare il file con lo stesso nome di prima, mentre se volessimo anche modificare il nome del file dovremo digitare "`:sav nuovo_nome_file`".Può capitare che ad esempio si modifichi un file che non doveva essere modificato, in questo caso possiamo uscire senza salvare; Per fare questo è necessario utilizzare l'operatore " `!` " che forza i comandi e ignora i messaggi d'errore come segue:

```
:q!
```

Una scorciatoia alla procedura di salvataggio e uscita spiegata prima è la seguente:

```
zz
```

Praticamente questo comando salva il file (che deve essere un file già esistente) ed esce.

Quindi si passa dalla sequenza:

```
ESC
:
wq
INVIO
```

Alla sequenza

```
ESC
zz
```

molto più veloce.

8.1.3 Comandi per muoversi

Fin qui abiamo visto come vim può essere utilizzato come un semplice editor di testo, ora vedremo i comandi per muoversi all'interno di un file.Ricordatevi di essere in modalità comando quando eseguite i seguenti comandi, la tabella elenca i principali comandi di vim.

TASTI	MOVIMENTO DEL CURSORE
h	Spostamento a sinistra di un carattere
l	Spostamento a destra di un carattere
k	Spostamento in su di una riga
j	Spostamento in giu di una riga
w	Spostamento in avanti di una parola
b	Spostamento in dietro di una parola
0	Spostamento all'inizio della riga
$	Spostamento alla fine della riga
INVIO	Spostamento all'inizio della riga precedente
(	Sposta il cursore all'inizio della frase
)	Sposta il cursorealla fine della frase;ripetendo il comando si sposta il cursore all'inizio della frase successiva

{	Sposta il cursore all'inizio del paragrafo
}	Sposta il cursore alla fine del paragrafo
CTRL-F	Sposta in avanti di uno schermo di testo;viene visualizzato lo schermo di testo successivo
CTRL-B	Sposta indietro di uno schermo di testo;viene visualizzato lo schermo di testo precedente
G	Sposta il cursore sull'ultima riga di testo
numG	Sposta il cursore su un numero di riga specificato: 45G posiziona il cursore sulla riga 45

TASTO	MOVIMENTO DEL CURSORE
INSERIMENTO	Tutti i comandi di inserimento attivano la modalità di inserimento;per uscire, premere il tasto ESC.
a	
A	
i	
I	
o	
O	

Capitolo 9

Il kernel GNU/Linux

Dopo aver parlato di Hurd, un capitolo dedicato ad uno dei kernel liberi più famosi in assoluto. In questo capitolo tutti i misteri di Linux.
Il Kernel Linux viene inventato da Linus Torvalds nel 1991. Visti i ritardi nello sviluppo del sistema Hurd (basato sul mikrokernel mach), Torvalds, come dice lui stesso "per divertirsi", programmò il suo kernel monolitico "quasi per caso". Il punto di forza del Kernel è sicuramente la comunità che si creò. Ogni appassionato diede il suo contributo allo sviluppo di quello che stava per diventare un sistema operativo a tutti gli effetti. Nel 1992 Linus Torvalds decise di distribuire il suo progetto con licenza GPL. La free software foundation, che proprio non riusciva a rendere stabile il suo Hurd, adottò il kernel Linux per il suo sistema operativo GNU: nacque GNU/Linux.

9.1 Le fasi del boot del nostro PC

9.1.1 Introduzione

Credo che per il primo articolo della rivista riguardante il Kernel Linux, non ci sia argomento migliore se non iniziare dall'inizio.
Cosa succede quando accendiamo la nostra macchina? In questo articolo voglio presentare una panoramica più o meno dettagliata di quello che succede dal momento in cui premiamo il pulsante di accensione del nostro PC, fino al cursore lampeggiante sul nostro terminale. Tutto questo si chiama fare il "booting" del PC.
La prima fase di avvio di un PC si definisce con il nome di bootstrapping.

9.1.2 Cenni storici

Il termine *boot* deriva dall'abbreviazione di *bootstrap* e storicamente è, metaforicamente parlando, preso dalla linguetta cucita posteriormente sugli stivali di pelle che permetteva a una persona di indossare i propri stivali senza un aiuto esterno[1].
Negli anni 50 premere il tasto di *bootstrap* di una macchina significava far leggere a un programma cablato, un programma di *bootstrap* da una scheda perforata senza un ulteriore aiuto dell'operatore.
Sebbene altre macchine precedentemente avessero iniziato ad usare il termine boot, la data più accreditata per i sistemi Unix è quella del 1971 con la prima edizione di "The Unix Programmer's Manual" del 3 Novembre.
Il termine *bootstrap* viene usato per la prima volta da IBM nel 1952 per le sua macchina 701. Con un bottone di accensione questa macchina leggeva la prima parola a 36 bit da una scheda perforata, da un lettore di schede, da un nastro magnetico o una unità a cilindri (predecessore del nostro odierno hard disk); metà parola veniva poi utilizzata per eseguire una istruzione che caricava altre istruzioni in memoria.

9.1.3 Le fasi del boot

L'avvio di un PC si esegue, a causa delle retrocompatibilità, ancora in un old-fashioned mode, cioè in vecchio stile e i passi di avvio di Linux (i primi due punti sono comunque

[1] Booting From Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Boot_loader

comuni, indipendentemente dal sistema operativo) in una macchina x86 si possono riassumere in questi punti[2]:

1. Il BIOS si avvia e cerca il primo dispositivo che ha attivo il flag di boot.
2. Carica il settore di boot chiamato bootsector in una specifica locazione di memoria.
3. Bootsector (o un altro bootloader come LILO o GNU GRUB) carica setup, alcune routine per la decompressione e l'immagine del kernel compressa.
4. il kernel viene decompresso in modalità protetta.
5. Inizializzazione a basso livello viene fatta da routine in assembly.
6. Inizializzazione di alto livello in C.

Questo il boot in a *nutshell*, vediamo ora di fare una panoramica più approfondita se non nel firmware del BIOS, non ancora disponibile al momento su tutti i pc (il progetto *OpenBIOS* si prefigge di creare una versione free per il firmware del BIOS [3]), almeno su quella del codice di GNU/Linux iniziando a vedere cosa fa un bootloader.

9.1.4 Cos'è un bootloader?

Quando un PC (i386) si avvia, la prima cosa che il BIOS fa è quella di leggere l'istruzione che si trova all'indirizzo `FFFF:0000`, che corrisponde fisicamente alla memoria `0xFFFF0` (all'avvio i registri sono inizializzati a questi valori `%ds=%es=%fs=%gs=%ss=0, %cs=0xFFFF0000, %eip = 0x0000FFF0`). Questa locazione, che si trova vicino alla fine dell'area di memoria di sistema (per le macchine a 32 bit si intende), contiene un jump al programma di startup del BIOS che dopo aver fatto dei check (POST *Power On Self Test*), cerca il dispositivo di boot che contiene nel suo primo settore la signature di boot `0xAA55` (vedremo in seguito del codice scritto in C per creare un settore di boot).
Il bootloader è il programma che carica il kernel del nostro sistema operativo. Un bootloader normalmente è diviso in due fasi, la fase 1 e la fase 2 (in alcuni casi, come *GRUB*, esiste anche una fase intermedia chiamata 1.5).
Nel caso del bootloader di Linux (almeno fino alla versione 2.4) la fase 1 è quella che si occupa di caricare il codice che dovrà a sua volta caricare il kernel, mentre nel caso di

[2]Campaign for Free BIOS http://www.fsf.org/campaigns/free-bios.html
[3]Tigran Aivazian - Linux Kernel Internals 2.4 http://www.moses.uklinux.net/patches/lki.html

altri bootloader, la fase 1 caricherà il codice del corrispettivo bootloader che ci permetterà poi di caricare il nostro kernel preferito. Questo codice si trova nel primo settore di 512 bytes e non può superare i primi 446 bytes a causa dei restanti 64 bytes che definiscono la partition table (più i 2 bytes di signature). Questo settore è comunemente conosciuto come MBR: *Master Boot Record*. La fase 2 di un bootloader caricherà il kernel e le routine di decompressione (naturalmente stiamo parlando del caso di un kernel Linux) fino a chiamare la nostra `init` che avvierà il sistema.
Anche se il kernel 2.6 ha abbandonato per ovvi motivi la prima fase del suo bootloader affidandosi a GRUB o LILO, presenterò qui una panoramica del boot concentrandomi sul bootloader di Linux, dando un occhio al primo codice e poi alla versione del kernel 2.6.

9.1.5 I bootloader di Linux

I bootloader per Linux più conosciuti sono sicuramente LILO e GRUB. Come vedremo in seguito il protocollo di boot di Linux ha un identificativo che il bootloader deve passare in fase di setup del kernel.
Ecco qui una lista di bootloader capaci di avviare Linux [4]:

1. LILO
2. Loadlin
3. bootsect-loader
4. SYSLINUX
5. EtherBoot
6. ELILO
7. GRUB
8. U-BOOT
9. Xen

[4] Feiyun Wang - Linux-i386-Boot-Code-HOWTO http://tldp.org/HOWTO/Linux-i386-Boot-Code-HOWTO/setup.html

Presumibilmente (e immancabilmente) ce n'è uno, o meglio c'era fino alla versione 2.4 del kernel, *bootsect-loader* (credo), che è proprio quello di Linux stesso. Questo articolo mostra parte del codice del bootloader di Linux con qualche accenno storico alla versione 0.0.1 di Linux che servirà per introdurre poi il codice del kernel 2.6.

9.1.6 Uno sguardo storico al primo codice

Permettetemi questa parafrasi: All'inizio fu boot.s. boot.s contiene il codice per il bootloader che avvierà il sistema e si preoccuperà di caricare in seguito head.s. Nel 1991 Linus Torvald aveva un HD di 40MB (basta guardare in config.h), 8MB di memoria e di sicuro un floppy drive da cui faceva il boot.
Presento qui delle parti di codice di boot.s (nei kernel più recenti si chiama bootsector.S), presenterò solo qui la parte di bootloader, il resto sarà dedicato al vero startup del kernel 2.6 (il lettore che vuole approfondire lo può fare leggendo direttamente l'intero codice dal sorgente):

```
BOOTSEG = 0x07c0 INITSEG = 0x9000
SYSSEG = 0x1000 | system loaded at 0x10000 (65536)
ENDSEG = SYSSEG + SYSSIZE
```

Qui sopra (esattamente come inizia `boot.s`) vengono definite le costanti degli indirizzi di memoria. La prima cosa che il BIOS fa quando trova un dispositivo di boot (in base alla signature) è quella di muovere i primi 512 bytes all'indirizzo di *BOOTSEG*.

```
entry start
start:
mov ax,#BOOTSEG
mov ds,ax | ds,es different data segments registers
mov ax,#INITSEG
...
```

le prime linee di codice di boot.s spostano il codice del bootloader all'indirizzo `#INITSEG`. Il registro di stack `%ss` viene impostato a `$INITSEG`.

```
mov ss,ax mov sp,#0x400 | arbitrary value >>512
```

si noti come lo *stack pointe*r viene valorizzato a un valore arbitario maggiore di 512 bytes. Siamo realmente all'inizio della fase di boot, ricordiamoci che non siamo in protected-mode ma ancora in real address, la memoria è completamente lineare.

Una volta che il codice è stato spostato a `#INITSEG`, il codice stampa il messaggio di "`Loading system ....`"

```
msg1:
.byte 13,10
.ascii "Loading system ..."
.byte 13,10,13,10
```

Bene il codice del bootloader si trova in memoria e ora deve caricare il sistema e "qualcuno" lo dovrà avviare.

Prima di procedere, qui troviamo una delle prime differenze tra il primo kernel e i kernel recenti. Se nella prima versione del kernel Linus mise insieme la parte di bootloader con la parte di "setup" (ovviamente era una distinzione inesistente ai tempi) che si occupa di passare in modalità protetta, di chiamare la init (`main.c`), ecc., ai giorni nostri chi si occupa di caricare in memoria la parte di setup e l'immagine del sistema è proprio il bootloader. Come vedremo in seguito, il bootloader in avvio caricherà il setup (`setup.S`) e l'immagine del sistema.

```
mov ax,#SYSSEG
mov es,ax | segment of 0x010000
call read_it
call kill_motor
```

`read_it` è la parte di codice che attiva il floppy e carica in memoria (in *SYSSEG*) i settori restanti sul disco che contengono il sistema (più avanti capiremo come la memoria viene mappata).

```
read_it: ...
| ENDSEG = SYSSEG + SYSSIZE
cmp ax,#ENDSEG | have we loaded all yet?
jb ok1_read
ret
```

ENDSEG non appare da nessuna parte nel codice di boot.s. La dimensione del sistema viene calcolata nel Makefile ad anteriori:

```
boot/boot: boot/boot.s tools/system
(echo -n "SYSSIZE = (";ls -l tools/system | grep system \
| cut -c25-31 | tr '\012' ' '; echo "+ 15 ) / 16") > tmp.s
cat boot/boot.s >> tmp.s
$(AS86) -o boot/boot.o tmp.s
rm -f tmp.s
$(LD86) -s -o boot/boot boot/boot.
```

Una volta caricato il codice in memoria se non ci sono problemi kill_motor viene chiamata, il nome è autoesplicativo.
Il sistema viene poi spostato nell'area di memoria 0x1000 (cosa che, come detto, oggi viene fatta da Setup.S arch/386/boot/compressed/{head.S,misc.c} che chiama le routine di decompressione del kernel). La dimensione massima che l'immagine del kernel può avere (*zImage*), può essere al massimo di 512Kb (0x80000) [5].

```
| first we move the system to it's rightful place
       mov     ax,#0x0000
       cld                     | 'direction'=0, movs moves forward
do_move:
       mov     es,ax           | destination segment
       add     ax,#0x1000
       cmp     ax,#0x9000
       jz      end_move
       mov     ds,ax           | source segment
       sub     di,di
       sub     si,si
       mov     cx,#0x8000
       rep
       movsw
       j       do_move
```

[5] H. Peter Anvin - THE LINUX/I386 BOOT PROTOCOL /usr/src/linux/Documentation/i386/boot.txt

Dopo aver impostato i segmenti di memoria con la gdts i passa in modalità protetta:

```
| Well, now's the time to actually move into protected mode. To make
| things as simple as possible, we do no register set-up or anything,
| we let the gnu-compiled 32-bit programs do that. We just jump to
| absolute address 0x00000, in 32-bit protected mode.
        mov     ax,#0x0001      | protected mode (PE) bit
        lmsw    ax              | This is it!
        jmpi    0,8             | jmp offset 0 of segment 8 (cs)
```

a questo punto il sistema è pronto e il codice passa l'esecuzione a head.s che contiene il codice di startup 32 all'indirizzo assoluto `0x00000000`.
D'ora in avanti possiamo abbandonare la parte storica di Linux. I kernel recenti (dalla versione 2.6) non usano più il bootsector di Linux ma si affidano ai più potenti GRUB o LILO, per cui la parte che segue tratta il codice che si può trovare in un kernel 2.6 (il kernel 2.4 supporta ancora il boot da floppy ed infatti si può trovare il suo bootloader qui `arch/i386/boot/bootsect.S`). Lascio ai curiosi la lettura del codice che è leggermente più complessa di quella del primo Linux.

9.1.7 Kernel decompression e kernel space

Il lavoro di bootloader viene oggi lasciato ad altri bootloader come LILO o GNU GRUB che sono più versatili e che caricano in memoria setup.S (è qui che troviamo tante cose interessanti, fate riferimento alla nota 5 *THE REAL-MODE KERNEL HEADER*) e bvmlinux, tutto procede poi come descritto di seguito.
Supponendo di aver costruito una immagine di bzImage (`__BIG_KERNEL__` **Nota**: *bImage* è stata abbandonata a partire dal protocollo di boot 2.02)[6] ecco cosa troviamo nel Makefile in *linux/arch/i386/boot/Makefile*:

```
bzImage         $(OBJCOPY) compressed/bvmlinux compressed/bvmlinux.out
                tools/build -b bbootsect bsetup compressed/bvmlinux.out
                $(ROOT_DEV) \
                        > bzImage
```

[6] Alessandro Rubini - Booting the kernel http://www.linux.it/~rubini/docs/boot/boot.html

A questo punto setup.S esegue:

1. alcune inizializzazioni hardware (calcola la memoria estesa presente nel sistema, tastiera, MCA, hard disk e mouse)
2. verifica se il BIOS supporta l'APM
3. inizializza l'A20
4. Prepara per la modalità protetta *code32_start che* è inizializzato a `0x1000` per *zImage* o `0x100000` per *bzImage*. Il valore di *code32* verrà utilizzato passando il controllo a *linux/arch/i386/boot/compressed/head.S*.
5. Cambia in modalità protetta

A questo punto il controllo passa alla routine che esploderà l'immagine del kernel in memoria al primo mega (`0x100000`) e a questo punto head.S salterà al vero inizio del kernel (*arch/i386/kernel/head.S* che è l'inizio del kernel decompresso).
Riporto i campi dell'header di `setup.S` per mostrare come il bootloader e il setup interagiscano per passarsi le informazioni della *command line* o dell'indirizzo dove si trova l'`initrd`. `setup` si aspetta che alcuni valori siano valorizzati dal bootloader alla sua partenza:

```
Offset  Proto   Name            Meaning
/Size
0200/2  2.00+ jump            Jump instruction
0202/4  2.00+ header          Magic signature "HdrS"
0206/2  2.00+ version         Boot protocol version supported
0208/4  2.00+ realmode_swtch  Boot loader hook
020C/2  2.00+ start_sys       The load-low segment (0x1000) (obsolete)
020E/2  2.00+ kernel_version  Pointer to kernel version string
0210/1  2.00+ type_of_loader  Boot loader identifier
0211/1  2.00+ loadflags       Boot protocol option flags
0212/2  2.00+ setup_move_size Move to high memory size (used with hooks)
0214/4  2.00+ code32_start    Boot loader hook
0218/4  2.00+ ramdisk_image   initrd load address (set by boot loader)
021C/4  2.00+ ramdisk_size    initrd size (set by boot loader)
0220/4  2.00+ bootsect_kludge DO NOT USE - for bootsect.S use only
0224/2  2.01+ heap_end_ptr    Free memory after setup end
0226/2  N/A   pad1            Unused
0228/4  2.02+ cmd_line_ptr    32-bit pointer to the kernel command line
```

```
022C/4  2.03+ initrd_addr_max Highest legal initrd address
```

`kernel/head.S` fa alcune inizializzazioni (tra cui copia anche i parametri della command line) e la prima CPU chiama `start_kernel()` che si trova in *init/main.c* (che è scritta in C).
`init/main.c` (`__init start_kernel(void)`) chiama le seguenti routine per fare alcune operazioni (riporto le principali chiamate seguendone l'ordine):

1. Si assicura che solo una CPU abbia il controllo.
2. Esegue alcune operazioni hardware.
3. Stampa a video il banner di Linux.
4. . Inizializza i *trap*.
5. Inizializza gli *IRQ*.
6. Inizializza lo *scheduler*.
7. Inizializza i *timers*.
8. Inizializza i *softirq*.
9. Fa il parsing della linea di comando (che ricordo è stata copiata fino a qui).
10. Inizializza la *console*.
11. Inizializza il sottosistema di caricamento dei moduli.
12. Abilita gli *interrupt*.
13. Chiama `mem_init()` che calcola alcuni valori della memoria e stampa a video il messaggio "`Memory:  ...`".
14. `kmem_cache_init()`, inizializza la cache del kernel.
15. Chiama il `fork_init`.
16. Prepara alcuni buffer per il *VFS*, ecc.
17. Se supportato inizializza il procfs.

18. Fa il fork di *init* e inizializza il massimo numero di threads in base alla memoria disponibile.

19. Controlla se ci sono delle patch specifiche per il processore, bus, ecc. su cui il kernel è in esecuzione.

20. crea un kernel `thread init()` che esegue un programma passato via `init = boot` parametre o prova ad eseguire uno dei seguenti programmi */sbin/init*, */etc/init*, */bin/init*, */bin/sh*.

9.1.8 User space

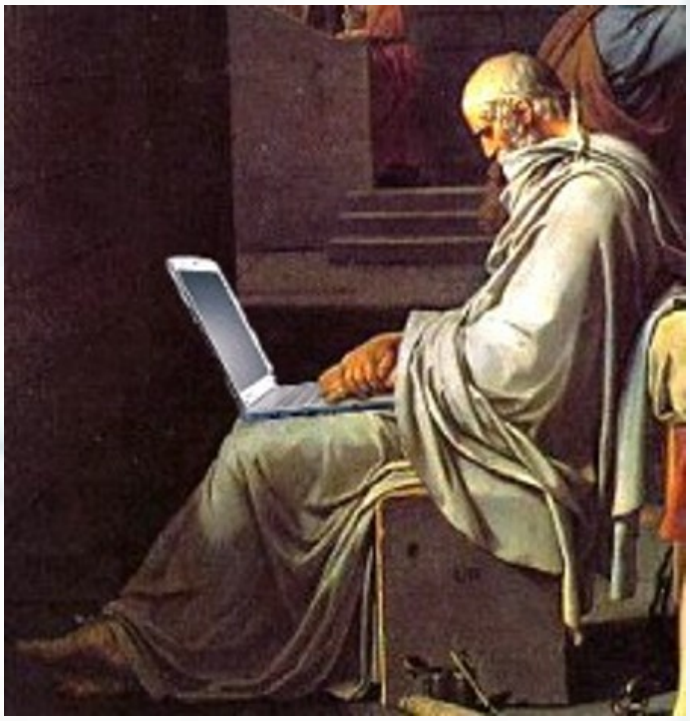
Arrivati a questo punto dopo la creazione del thread di *init* il kernel si mette nel *loop* di *idle* con il `pid=0`. Quello che succede dopo dipende dal nostro programma di init.
Il nostro sistema è ora pronto ad eseguire i nostri programmi.

9.1.9 Conclusioni

Molte cose sono cambiate dal lontano 1991, anche se sostanzialmente la struttura è rimasta pressoché invariata, nel boot di Linux ma è divertente vedere come i commenti originali di Linus Torvalds sono rimasti invariati e al pi ù dei nuovi se ne sono aggiunti man mano che lo sviluppo del codice aggiungeva nuovi componenti. Questo articolo è stato solo una piccola introduzione anche se ha sfiorato alcuni concetti abbastanza a fondo e spero di aver reso un'idea chiara delle fasi boot. Nella realtà uno studio completo del boot di Linux richiede molto tempo se si prende in considerazione un kernel odierno. Per questo motivo ho iniziato presentando il primo codice scritto da Torvalds per evitare di dilagare troppo, cosa che non ho potuto fare quando sono passato al codice recente. Chi è curioso e vuole approfondire lo può fare iniziando a seguire alcuni link riportati nell'articolo e a leggere dall'inizio il codice naturalmente.

Capitolo 10

Storia e filosofia del software libero



Nel primo capitolo, argomento uguale, campo diverso. Se debian è un ottimo sistema libero e sicuramente appartiene a questo tema, il software libero si estende ben al di là del *figlio* di Murdock. In questa sezione vi proporremo una serie di articoli relativi all'immenso mondo del software libero. Cercheremo di illustrarvi le basi del pensiero, così come l'evoluzione nel corso degli anni.

10.1 Informatica e Pubblica Amministrazione - Parte prima

10.1.1 Le basi: il codice sorgente aperto

Definizione di open-source

La legge Regionale Toscana 1/2004 - Art. 3 comma 1d, dà la seguente definizione di programma a codice sorgente aperto:

```
...programma per elaboratore la cui licenza di distribuzione consente all'utente
di accedere al codice sorgente per studiarne il funzionamento, apportarvi modifi-
che, mantenerlo nel tempo, estenderlo e ridistribuirlo".
```

La Free Software Foundation, attraverso quattro punti principali, dà la seguente definizione di software libero:

```
Libertà di eseguire il programma, per qualsiasi scopo.
```

Imporre restrizioni sull'uso del Software Libero, in termini di tempo (periodo di prova di 30 giorni, la licenza scade il 1 Gennaio 2004) o di scopo (il permesso è accordato per usi di ricerca o non commerciali, non può essere usato per fare benchmarking), o limitazioni arbitrarie di area geografica (non può essere usato nel paese X) rende un programma non libero.

```
Libertà di studiare come funziona il programma e adattarlo alle proprie necessità.
```

Anche imporre restrizioni di fatto o di diritto sulla comprensione o la modifica di un programma, ad esempio richiedendo l'acquisto di licenze speciali o la firma di un Non-Disclosure-Agreement (NDA) o, per i linguaggi di programmazione che sono rappresentabili in più forme, vietando l'accesso al mezzo più naturale per comprendere o modificare un programma (codice sorgente), lo rende proprietario (non libero). Senza la libertà di modificare un programma, la gente sarebbe alla mercè di un singolo fornitore.

```
Libertà di ridistribuire copie in modo da aiutare il prossimo.
```

Il software può essere copiato e distribuito praticamente senza costi: se non si ha il permesso di dare un programma a qualcuno che ne ha bisogno (anche dietro pagamento, se lo si vuole), il programma non è libero.

```
Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti,
in modo tale che tutta la comunità ne tragga beneficio.
```

Nessuno è un bravo programmatore in tutti i campi, qualcuno non sa programmare del tutto. Questa libertà permette a chi non ha il tempo o le capacità per risolvere un problema di accedere indirettamente alla libertà di modifica. Anche questo può avvenire dietro un compenso.

Definizione di Pubblica Amministrazione

Un'ottima definizione ci viene fornita da Wikipedia:

```
Nell'ordinamento italiano la Pubblica Amministrazione (P.A.) è un insieme di
enti e soggetti pubblici ( comuni,provincia, regione, stato, ministeri, etc. ) e
talora privati (organismi di diritto pubblico, concessionari,amministrazioni ag-
giudicatrici, s.p.a. miste), e tutte le altre figure che svolgono in qualche mo-
do la funzione amministrativa nell'interesse della collettività e quindi , nell'
interesse pubblico, alla luce del principio di sussidiarietà.
```

Tutto quanto detto fino a qui (definizioni, citazioni di leggi, ecc....) ci può aiutare a capire fin da subito quali dovrebbero essere i criteri per una buona informatizzazione, con software a codice sorgente aperto, delle Pubbliche Amministrazioni. Intanto, possiamo porci una domanda fondamentale e, per essa, trovare alcune possibili soluzioni. Perchè le Pubbliche Amministrazioni dovrebbero orientarsi verso software a codice sorgente aperto? Si possono dare, a questa domanda, molteplici risposte, tutte ugualmente valide:

- I dati devono essere di proprietà della struttura che li gestisce e, non solo devono essere sempre e immediatamente utilizzabili, ma la struttura di questi dati deve essere aperta, in modo che se si vuole cambiare fornitore software, non ci devono essere problemi di migrazione.

- Quindi non ci devono essere ostacoli tecnici che impediscano il cambio di fornitore.
- Il contenuto del software deve essere certo e lo stesso software deve fare solo quanto gli viene richiesto (eliminazione pericolo di backdoors).
- Il codice acquisito deve essere di proprietà dell'amministrazione acquirente.
- La Pubblica Amministrazione proprietaria degli applicativi che sono stati realizzati secondo specifiche richieste ed esigenze di quell'Amministrazione, deve avere la possibilità di cederli in uso gratuito ad altre amministrazioni, che con il minimo sforzo potranno adattarli alle proprie esigenze (utilizzo per esempio di repository comuni di facile accesso).
- La Pubblica Amministrazione deve mettere a disposizione degli utenti, dati (documenti, moduli, ecc..) immediatamente utilizzabili e, l'utente o il cittadino, non deve essere obbligato ad acquistare software proprietario per usufruire dei servizi on-line.

Quali difficoltà potrebbero avere le P.A. Acquistando software open source? In pratica non ci sono difficoltà e chi sviluppa oggi software proprietario per le Pubbliche Amministrazioni, potrebbe allo stesso modo sviluppare direttamente software open source, con garanzie e vantaggi immediati per le P.A.:

- proprietà del codice, dei dati e delle strutture
- modificabilità e adattabilità del software
- possibilità di ridistribuzione dello stesso software.

10.1.2 Analisi delle possibili difficoltà

Analizziamo nel dettaglio le possibili difficoltà (ammesso che ci siano) che le P.A. potrebbero incontrare nella migrazione verso software a codice sorgente aperto, suddividendo lo stesso software per tipologia di utilizzo:

- Software di base
- Software di produttività personale
- Software applicativo
- Software "custom"

Il software di base

Per Software di base si intendono non soltanto i Sistemi Operativi ma anche DBMS, Utility di sistema e di amministrazione, Web server, File and Print Server, Proxy, Firewall,... A questo livello c'è una grande diffusione di soluzioni di OSS anche nelle PAL. Comuni medio grandi e Province spesso usano infatti sistemi Open Source soprattutto per la gestione del Web Server, per la posta elettronica e per la gestione della sicurezza della rete (firewall).

Il software di produttività personale

Per Software di produttività personale si intendono Editori di testo, Fogli elettronici, Data Base e tutti gli applicativi non relativi a uno specifico settore e non verticali. In questo campo incominciano a vedersi le prime sperimentazioni di passaggio, ad esempio, da Office a OpenOffice, tuttavia restano ancora alcune difficoltà:

- non completa compatibilità tra i diversi pacchetti
- difficoltà di importazione e/o esportazione di dati da un software ad un altro e/o da un computer ad un altro.

Tale software sta diffondendosi sempre di più nei settori della Pubblica Amministrazione Locale che hanno minori necessità di importare ed esportare dati e con minore impatto sull'utenza o che comunque non sono settori di front-office.

Il software applicativo

A questo livello le difficoltà potrebbero essere minori, almeno da un punto di vista teorico, in quanto chi sviluppa software per la Pubblica Amministrazione potrebbe direttamente sviluppare software Open Source. In particolare è qui importante notare come, non solo le Pubbliche Amministrazioni Locali raramente richiedano che il software acquisito sia Open Source, ma come sia quasi totalmente disattesa la norma in base alla quale le Pubbliche Amministrazioni "titolare di programmi applicativi realizzati su specifiche indicazioni del committente pubblico, hanno facoltà di darli in uso gratuito ad altre amministrazioni pubbliche, che li adattano alle proprie esigenze" (Art. 25, comma 1, Legge 24 Novembre 2000 N° 340).

Il software "custom"

La proprietà del codice . I Capitolati devono prevedere che il codice sviluppato sia di proprietà dell'amministrazione appaltante (anche se non obbligatoriamente in forma esclusiva). La proprietà dei dati . Nei capitolati deve essere esplicitato che non solo i dati sono, ovviamente, di proprietà dell'amministrazione appaltante, ma anche la loro struttura. Questo vale sia per software Open Source che Proprietario!
La prima parte, introduttiva, si conclude quì. Nella seconda parte, che presumibilmente sarà pubblicata nel prossimo numero, cominceremo ad approfondire tutte queste tematiche, iniziando con un pò di storia degli anni 2000 e, proseguendo con la direttiva "Stanca" e la commissione "Meo".

Articolo redatto con il contributo della documentazione resa pubblicamente disponibile, sotto licenza CC, dalla dottoressa Flavia Marzano (marzano@cibernet.it)

Impressum

Redattori articoli

- La nascita di debian - *brunitika*
- Installazione grafica di lenny - *borlongioffei* (introduzione), *xtow* (installazione)
- Installazione di debian GNU/Hurd su QEMU - *brunitika*
- Installazione lenny su eeePC 900A - *borlongioffei*
- Crittografia portami via - *ugaciaka*
- Fluxbox: installazione e configurazione - *hjubal*
- Fluxbox: configurazione su eeePC - *ugoboss*
- Guida vim - *chris*
- Le fasi del boot del nostro PC - *galileo75*
- Informatica e pubblica amministrazione - *marbel*

Revisori articoli

mm-barabba, *simone*, *brunitika*, *borlongioffei*, *xtow*, *marbel*

Copertina

mm-barabba

Impaginazione

borlongioffei (versione stampa), *brunitika* (web-zine)

Collaboratori

bel.gio, *gippasso*, *maxer*, *marbel*

Contatto

Tutti i membri del progetto sono reperibili sul forum del portale *www.debianizzati.org*, dove è possibile trovarci cercando l'utente relativo nel forum.

Nota a questa versione stampabile

Quella che state leggendo è la versione stampabile della *e-zine* "Debianizzati" prodotta dalla comunità *www.debianizzati.org*.
Potete trovare la versione on-line, comodamente consultabile con il proprio browser, all'indirizzo http://e-zine.debianizzati.org/.

I sorgenti LaTeX di quesa versione sono disponibili all'indirizzo sopra riportato.

`Happy Debian, Happy hacking`